CSE 132A: Database System Principles
Summer Session I, 2013

Midterm (A)

(Thursday, July 18)

Total: 16 points

Problem 1 *(3 points).* For each of the following statements **circle** the correct answer, indicating whether it is true of false (no justification required):

(a) If A is an attribute of type integer in a table, then A*0 always evaluates to true.

TRUE **FALSE**

Explanation: If A IS NULL, then A*0 evaluates to UNKNOWN.

(b) In SQL, all nested queries can be unnested.

TRUE **FALSE**

Explanation: SQL queries with NOT IN cannot be unnested (see SQL DML slideset, slides #37-41).

(c) In SQL, the x > ALL(subquery) where x is an attribute name can be simulated through NOT IN.

TRUE FALSE

Explanation: Assume that 'subquery' returns column y. Then 'x > ALL(subquery)' can be translated to 'x NOT IN (SELECT x FROM subquery WHERE x <= y)'.

(d) In SQL, the EXISTS clause can be simulated with the COUNT function.

TRUE FALSE

Explanation: 'EXISTS(subquery)' becomes '(SELECT COUNT(*) FROM subquery) >0'.

(e) In SQL, the following two queries Q_1 and Q_2 over the relation person(name: string, age: integer) *always* return the same result:
Q_1: SELECT * FROM person
Q_2: SELECT * FROM person WHERE age <= 30 OR age >30

TRUE **FALSE**

Explanation: Q_1 returns all tuples from the relation person, while Q_2 returns only those tuples that have a non-null age value.

(f) Given relation R(x: integer, y: integer), where x is the primary key of R, the query "SELECT x FROM R" always returns a **set** of tuples (i.e., it does not return duplicates).

TRUE FALSE

Explanation: Since x is the primary key of R, no two tuples in R will have the same x value.

(g) If an attribute A of relation R is a foreign key referencing attribute B of relation S, then attribute B should be the **primary key** of S.

TRUE **FALSE**

Explanation: According to the slides and the textbook, a referenced attribute has to be a primary key. However, as we said in class, DBMSs also allow a referenced attribute to be a (non-primary) candidate key. Therefore, either choice will be considered correct.

(h) SQL is a procedural language.

TRUE **FALSE**

Explanation: SQL is a declarative language; we specify *what* we want & *not how* to compute it.

(i) If Person(ID: integer, Name: string, Address: string) is a relation with primary key the attribute ID, then the query "SELECT * FROM Person" returns the tuples in ascending order of their ID.

TRUE **FALSE**

Explanation: Unless we use the ORDER BY keyword in a SQL query, its result is unordered (i.e., it could appear in any order).

(j) Given relation R(x: integer, y: integer), the following is a valid SQL query:

```
SELECT s.x
FROM R s, R t
WHERE s.x > s.y
```

TRUE FALSE

(k) Given relation R(x: integer, y: integer), the following is a valid SQL query:

```
SELECT x, y
FROM R
WHERE x > 5
GROUP BY x
```

TRUE **FALSE**

Explanation: If a query contains a GROUP BY clause, in its SELECT clause it can have only attributes that appear in the GROUP BY clause or aggregate functions over the rest of the attributes. In this case, y does not satisfy either of the two conditions.

(l) In SQL, the ORDER BY clause can be simulated by the COUNT function.

TRUE **FALSE**

Explanation: Only the ORDER BY clause can impose an ordering on the query results.

Problem 2 *(1 point).* For what values of x and y, including NULL, does the Boolean expression $x > 5$ *AND NOT*$(y <= 2)$ return the truth value UNKNOWN in SQL?

Solution:
The expression evaluates to UNKNOWN when:
(x IS NULL AND y IS NULL) OR
($x > 5$ AND y IS NULL) OR
(x IS NULL AND $y > 2$)

Problem 3 *(2 points.)* Gradiance homework: The table Scores(Team, Day, Opponent, Runs) gives the scores in the Japanese Baseball League for two consecutive days. The data in this table is as follows:

Team	Day	Opponent	Runs
Dragons	Sunday	Swallows	4
Tigers	Sunday	Bay Stars	9
Carp	Sunday	Giants	2
Swallows	Sunday	Dragons	7
Bay Stars	Sunday	Tigers	2
Giants	Sunday	Carp	4
Dragons	Monday	Carp	6
Tigers	Monday	Bay Stars	5
Carp	Monday	Dragons	3
Swallows	Monday	Giants	0
Bay Stars	Monday	Tigers	7
Giants	Monday	Swallows	5

Consider the following query:

```
SELECT Team, Day
FROM Scores S1
WHERE Runs <= ALL
      (SELECT Runs FROM Scores S2
      WHERE S1.Day = S2.Day)
```

(a) Write the result of the query (i.e., the table returned when the query is evaluated) on the above table.

(b) Explain in natural language what this query returns. It should not be an explanation of how the query works or how it is structured. Instead it should be an intuitive explanation to somebody that does not know or want to learn databases, similar to the explanation "Return actors that like every movie by 'Berto'" we saw in class.

Solution:
(a) The query returns the following result:

Team	**Day**
Carp	**Sunday**
Bay Stars	**Sunday**
Swallows	**Monday**

(b) The query returns the teams that had the lowest number of runs on a given day among all teams that played on that day together with that day.

Problem 4 *(8 points.)* Consider the database of a car rental company with the following schema:

driver(<u>driverName</u>, age)
car(<u>VIN</u>, make, model, color)
rental(<u>driverName, VIN, date</u>)

The relation *driver* lists the name and age of drivers, the relation *car* lists the VIN (vehicle identification number), make, model and color of cars and the relation *rental* contains the name of the driver that rented some car identified by its VIN on a given date. The underlined attributes form the primary key of the corresponding relation. In the case of *rental* all its attributes form its primary key. Additionally, *rental.driverName* is a foreign key referencing *drive.driverName* and *rental.VIN* is a foreign key referencing car.VIN. Here is an example instance over this schema:

driver	*driverName*	*age*
	Schumacher	44
	Alonso	31
	Vettel	26

car	*VIN*	*make*	*model*	*color*
	SARW	Mercedes	SLS	silver
	FRRI	Ferrari	Enzo	red
	ULTM	BMW	1M	red
	LMS	Mercedes	C280	red
	CRVT	Chevrolet	Corvette	yellow
	GTR	Nissan	GTR	silver

rental	*driverName*	*VIN*	*date*
	Schumacher	SARW	07-15-2013
	Schumacher	SARW	01-01-2012
	Schumacher	LMS	01-02-2012
	Alonso	FRRI	05-05-2013
	Vettel	SARW	02-02-2012
	Vettel	GTR	12-12-2012

For each of the following statements write a SQL statement that has the corresponding effect. The query should of course work on any database instance over the schema outlined above (i.e., not only on the above example instance). *For your answers use the space below each question or on the back of the page.*

(a) List distinct names of drivers (i.e., without duplicates) that have rented at least one Mercedes (in this case Mercedes is the car make).

Solution:
```
SELECT DISTINCT driverName
FROM rental, car
WHERE rental.VIN = car.VIN AND make = 'Mercedes'
```

(b) List the names of drivers that have rented **every red car**. Note that the question is not asking for the drivers that have rented **only** red cars. A driver will be part of the output as long as he has rented every red car (even if he has rented some other non-red car).

Solution:
```
SELECT driverName
FROM driver d
WHERE NOT EXISTS
```

```
(SELECT VIN
FROM car c
WHERE color = 'red' AND NOT EXISTS
      (SELECT VIN
      FROM rental r
      WHERE r.driverName = d.driverName AND r.VIN = c.VIN))
```

(c) For each car color, list the number of cars of that color. The output should be a set of (color, numberOf-Cars) pairs.

Solution:

```
SELECT color, COUNT(*)
FROM car
GROUP BY color
```

(d) Delete from relation *driver* all drivers over the age of 40 that have rented a Chevrolet Corvette (in this case the make is Chevrolet and the model is Corvette).

Solution:

```
DELETE FROM rental WHERE driverName IN
      (SELECT driverName
      FROM driver
      WHERE age > 40 AND driverName IN
            (SELECT driverName
            FROM car, rental
            WHERE car.VIN = rental.VIN AND make = 'Chevrolet' AND model = 'Corvette')
)

DELETE FROM driver
WHERE age > 40 AND driverName IN
      (SELECT driverName
      FROM car, rental
      WHERE car.VIN = rental.VIN AND make = 'Chevrolet' AND model = 'Corvette')
```

Problem 5 *(2 points.)* Consider the database of an apartment complex containing the following relation:

apartment(unitNumber: integer, size: integer, vacant: boolean)

In this relation unitNumber is the primary key. Each tuple in this relation specifies the number of a unit, its size in square feet and whether it is vacant or not. For example, the tuple (112, 650, true) specifies that the unit with number 112 has size 650 sq ft and is vacant.

Write a query that lists all **vacant** apartments (identified by their unitNumber) that have the second-largest size **among all vacant apartments** together with this size. Please take the following instructions into account when writing your query:

- You should consider the second-largest size of **vacant apartment only**. To find the second-largest size you should ignore the sizes of non-vacant apartments.
- There could be multiple vacant apartments that are have the **same** largest size among all vacant apartments. In that case, your query should return all vacant apartments that have the immediately smaller size.

For instance, given the following relational instance:

apartment	unitNumber	size	vacant
	100	600	true
	101	800	false
	102	700	false
	103	700	true
	104	700	true
	105	600	true
	106	500	true

the query should return:

unitNumber	size
100	600
105	600

Solution:

```
SELECT unitNumber, size
FROM apartment
WHERE vacant AND size =
      (SELECT MAX(size)
      FROM apartment
      WHERE vacant AND
            size <> (SELECT MAX(size) FROM apartment WHERE vacant))
```