CSE-3421M Test #2

Queries

Sur / Last Name:
Given / First Name:
Student ID:

- **Instructor:** Parke Godfrey
- **Exam Duration:** 75 minutes
- **Term:** Winter 2014

Answer the following questions to the best of your knowledge. Your answers may be brief, but be precise and be careful. The exam is closed-book and closed-notes. Calculators, etc., are fine to use. Write any assumptions you need to make along with your answers, whenever necessary.

There are four major questions, each with parts. Points for each question and sub-question are as indicated. In total, the test is out of 50 points.

In schemas, the underlined attributes denote a table's key. Attributes that are in *italics* are not nullable. Foreign keys are indicated by FK.

If you need additional space for an answer, just indicate clearly where you are continuing.

Marking Box	
1.	/10
2.	/15
3.	/15
4.	/10
Total	/50

1. (10 points) **Relational Algebra & Calculus.** *Don't divide by zero!* [Analysis]

   Consider table **R** with two attributes A and B.

$$(\pi_{\mathsf{A}}(\sigma_{\mathsf{B}<5}(\mathbf{R}))) \bowtie (\pi_{\mathsf{A}}(\sigma_{\mathsf{B}\geq 5}(\mathbf{R})))$$

   a. (3 points) Rewrite the relational-algebra expression above as an equivalent SQL query.

```
select X.A
from R X, R Y
where X.A = Y.A
  and X.B < 5
  and Y.B >= 5;
```

   b. (2 points) Say that you further knew $\mathbf{R}(\underline{\mathsf{A}}, B)$; that is, A is the primary key of **R** and **B** is not nullable. Dr. Dogfurry says the relational-algebra expression above will then evaluate to the *empty* answer set, regardless of what is in table **R**!

   Either explain why he is correct, *or* construct a small example of **R**—tuples in **R**—that would result in an answer.

He is correct. The predicates $B < 5$ and $B \geq 5$ partition the tuples into two disjoint sets. As A is the primary key, there can be no A value shared between the two sets.

For Questions 1c and 1d, consider the schema $\mathbf{R}(\underline{\mathsf{A}}, \underline{\mathsf{B}})$, and $\mathbf{S}(\underline{\mathsf{B}}, \underline{\mathsf{C}})$.

c. (3 points) Rewrite the relational-calculus expression

$$\{\langle A, C\rangle \mid \exists \mathsf{B}(\langle A, B\rangle \in \mathbf{R} \wedge \langle B, C\rangle \in \mathbf{S})\}$$

as a succinct, equivalent *relational-algebra* expression.

$\pi_{\mathsf{A},\mathsf{C}}(\mathbf{R} \bowtie \mathbf{S})$

d. (2 points) Rewrite the relational-calculus expression

$$\{\langle A, C\rangle \mid \exists \mathsf{B}(\langle A, B\rangle \in \mathbf{R}) \wedge \forall \mathsf{B}(\langle A, B\rangle \in \mathbf{R} \rightarrow \langle B, C\rangle \in \mathbf{S})\}$$

as a succinct, equivalent *relational-algebra* expression.

We do not have anything like '∀' in relational algebra (except divide*, which captures a part of it). So we have to carefully think about what the relational calculus (RC) expression is saying, and how to translate it.*
The RC query is saying report each $\langle \mathsf{A}, \mathsf{C}\rangle$ *such* A *appears in* $\mathbf{R}$ *(*$\langle \mathsf{A}\rangle \in \mathbf{R}$*),* C *appears in* $\mathbf{S}$*, and, for every* $\langle \mathsf{A}, \mathsf{B}\rangle$ *in* $\mathbf{R}$ *for that* A*,* $\langle \mathsf{B}, \mathsf{C}\rangle$ *is in* $\mathbf{S}$*.*

$\pi_{\mathsf{A},\mathsf{C}}(\mathbf{R} \bowtie \mathbf{S}) - \pi_{\mathsf{A},\mathsf{C}}((\mathbf{R} \times \pi_{\mathsf{C}}(\mathbf{S})) - (\mathbf{R} \bowtie \mathbf{S}))$

2. (15 points) **SQL.** *Singular Quantum Leap.* [Exercise]

Person(p#,*name*,birthdate,*nationality*,gender)
Actor(p#,*aguild#*)
FK (p#) refs **Person**
Director(p#,*dguild#*)
FK (p#) refs **Person**
Writer(p#, *wguild#*)
FK (p#) refs **Person**
Studio(name)
ScreenPlay(title,year)
Authored(title,year,writer)
FK (title, year) refs **ScreenPlay**
FK (writer) refs **Writer** (p#)
Movie(title,studio,year,*genre*,*director*,*length*)
FK (studio) refs **Studio** (name)
FK (title, year) refs **ScreenPlay**
FK (director) refs **Director** (p#)
Cast(title,studio,year,role actor,*minutes*)
FK (title, studio, year) refs **Movie**
FK (actor) refs **Actor** (p#)
Affiliated(director,studio)
FK (director) refs **Director** (p#)
FK (studio) refs **Studio** (name)

Figure 1: Movie Schema.

The basic schema of a database for tracking movies, actors, directors, and screenplay writers is shown in Figure 1. The underlined attributes indicate a table's primary key (and are hence not nullable). Additionally, attributes that are in *italics* are not nullable. Foreign keys are indicated by FK.

Additional implicit constraints on the database are that a director should be affiliated with at least one studio, and a screenplay ought to have at least one author.

a. (5 points) Write an SQL query that lists *writers* (**Writer**) and *movies* (**Movie**)—by p#, name, nationality, role, title, studio, and year— for which the writer also was an *actor* (**Actor**) in that movie (**Cast**).

```
select P.p#, P.name, P.nationality,
       C.role, C.title, C.studio, C.year
from Person P, Cast C, Authored A
where  C.title = A.title  -- C's a writer of the screenplay
  and   C.year = A.year
  and  C.actor = A.writer
  and A.writer = P.p#;    -- Join writer and person
```

b. (5 points) Write an SQL query that lists *movies* (**Movie**)—by title, studio, year, director, and the director's name—for which the movie's *director* is *not* affiliated (**Affiliated**) with the movie's studio (**Studio**).

```
select M.title, M.studio, M.year, M.director, P.name
from Movie M, Person P
where M.director = P.p#
  and M.studio not in (
        select A.studio
        from Affiliated A
        where A.director = M.director
    );
```

c. (3 points)

State in *plain, concise English* what the following SQL query does.

You get *zero* credit if you use database terms in your answer! (E.g., "Well, the query first *joins* two tables, taking the *projection* of..." does not count!)

```
with
    Appeared (p#, title, year, studio) as (
        select distinct C.actor, C.title, C.year, C.studio
        from Cast C
    )
select distinct A.p#, A.name, B.p#, B.name
from Person A, Person B, Appeared AM, Appeared AN, Appeared BM, Appeared BN
where A.p# = AM.p# and B.p# = BM.p#
  and AM.title = BM.title and AM.year = BM.year and AM.studio = BM.studio
  and A.p# = AN.p# and B.p# = BN.p#
  and AN.title = BN.title and AN.year = BN.year and AN.studio = BN.studio
  and (AM.title <> AN.title or AM.year <> AN.year or AM.studio <> AN.studio)
  and A.p# < B.p#;
```

State pairs of actors by ID and name who have acted in at least two movies together.

---

d. (2 points) If we rewrote the query in Question 2c just to source **Cast** in the from of the main query instead of using the sub-query **Appeared**, would it necessarily evaluate to the same answers as the query in Question 2c or not?

Briefly explain.

Assuming we change "p#" to "actor" in the where *clause, ut would be the same. The* distinct *assures we do not list a pair twice;* **Appeared** *is not needed for that.*
And "role" does not matter in this case as we explicitly find two different movies the people were in together. If we did this by aggregate counting, however, "role" could mess us up if we were not careful!

3. (15 points) **General.** *All the beautiful choices!* [Multiple Choice]

Choose *one* best answer for each of the following. Each is worth one point. There is no negative penalty for a wrong answer.

In the rare case you feel a clarification to your answer is needed, write a brief clarification on the side.

---

a. Why are NULL values needed in the relational model? NULL values can be used for all but which one of the following?

A. To avoid confusion with actual legitimate data values like 0 for integer columns and ' ' (the empty string) for string columns.

B. To allow duplicate tuples in the table by filling the primary key column(s) with NULL.

C. To leave columns in a tuple marked as "unknown" when the actual value is unknown.

D. To fill a column in a tuple when that column does not really "exist" for that particular tuple (e.g., 'Parke' has no telephone).

E. To opt a tuple out of enforcement of a foreign key.

For Questions 3b and 3c, consider the table $\mathbf{R}(\underline{A}, B)$ which is *empty* (that is, it has no tuples).

---

b. What does the query

```
select max(R.B) from R;
```

return?

A. An empty table of one column.

B. A table of one column with one row with the value $\langle \text{NULL} \rangle$.

C. A table of one column with one row with the value $\langle \text{INF} \rangle$.

D. An error message.

E. *Not enough information to determine.*

---

c. What does the query

```
select R.A, max(R.B) from R group by R.A
```

return?

A. An empty table of two columns.

B. A table of columns with one row with the value $\langle \text{NULL}, \text{NULL} \rangle$.

C. A table of two columns with one row with the value $\langle \text{NULL}, \text{INF} \rangle$.

D. An error message.

E. *Not enough information to determine.*

---

d. Consider table $\mathbf{R}(\underline{K}, A, B)$. Which of the following SQL queries is illegal?

A. `select A from R;`

B. `select K, count(*) from R;`

C. `select A, count(*) from R group by A;`

D. `select B, count(*) from R group by A, B;`

E. `select A, B, count(*) from R group by A, B;`

For Questions 3e–3h, consider the schema

R(A, *B*) FK (B) refs **S**

S(*A*, B) FK (A) refs **R**

None of the attributes is nullable. **R** contains 75 tuples and **S** contains 25 tuples.

e. What is the *least* number of tuples that $\mathbf{R} \bowtie \mathbf{S}$ contains?

A. 0
B. 1
C. 25
D. 75
E. 1,875

f. What is the *most* number of tuples that $\mathbf{R} \bowtie \mathbf{S}$ contains?

A. 0
B. 1
C. 25
D. 75
E. 1,875

g. What is the *least* number of tuples that $\sigma_{\mathsf{A}=7}(\mathbf{R}) \bowtie \mathbf{S}$ contains?

A. 0
B. 1
C. 25
D. 75
E. 1,875

h. What is the *most* number of tuples that $\sigma_{\mathsf{A}=7}(\mathbf{R}) \bowtie \mathbf{S}$ contains?

A. 0
B. 1
C. 25
D. 75
E. 1,875

For Questions 3i–3l, one of the choices is not like the others; that is, one of the choices could evaluate to a different answer than the others do.

Choose the one that may evaluate differently.

---

i. Consider tables $\mathbf{R}(\underline{\mathsf{A}}, \underline{\mathsf{B}})$ and $\mathbf{S}(\underline{\mathsf{A}}, \underline{\mathsf{B}})$.

A. $\mathbf{R} \cap \mathbf{S}$
B. $\mathbf{R} - (\mathbf{R} - \mathbf{S})$
C. $(\mathbf{R} \cup \mathbf{S}) - ((\mathbf{R} - \mathbf{S}) - (\mathbf{S} - \mathbf{R}))$
D. $((\mathbf{R} \cup \mathbf{S}) - (\mathbf{R} - \mathbf{S})) - (\mathbf{S} - \mathbf{R})$
E. $\mathbf{R} \bowtie \mathbf{S}$

---

j. Consider the relations $\mathbf{R}(\underline{\mathsf{A}}, \underline{\mathsf{B}})$ and $\mathbf{S}(\underline{\mathsf{B}}, \underline{\mathsf{C}})$.[1]

A. $\pi_{\mathsf{A}}(\mathbf{R} \bowtie \mathbf{S})$
B. $\pi_{\mathsf{A}}(\mathbf{R}) - (\pi_{\mathsf{A}}(\mathbf{R}) - \pi_{\mathsf{A}}(\mathbf{R} \bowtie \mathbf{S}))$
C. $\pi_{\mathsf{A}}(\mathbf{R}) - (\pi_{\mathsf{A}}(\mathbf{R} - \pi_{\mathsf{A,B}}(\mathbf{R} \bowtie \mathbf{S})))$
D. $\pi_{\mathsf{A}}(\mathbf{R} \cap (\pi_{\mathsf{A}}(\mathbf{R}) \times \pi_{\mathsf{B}}(\mathbf{S})))$
E. $\pi_{\mathsf{A}}((\mathbf{R} \times \pi_{\mathsf{C}}(\mathbf{S})) \cap (\pi_{\mathsf{A}}(\mathbf{R}) \times \mathbf{S}))$

---

k. Consider the schema $\mathbf{R}(\underline{\mathsf{A}}, \underline{\mathsf{B}})$ and $\mathbf{S}(\underline{\mathsf{B}}, \underline{\mathsf{C}})$.

A.
```
select distinct R.A, S.C from R, S s1
       where R.B in (select s2.B from S s2 where s1.C = s2.C);
```
B.
```
select distinct R.A, S.C from R, S where R.B = S.B;
```
C. $\pi_{\mathsf{A,C}}(\mathbf{R} \bowtie \mathbf{S})$
D. $\{\langle A, C\rangle \mid \neg \forall B(\langle A, B\rangle \in \mathbf{R} \rightarrow \langle B, C\rangle \notin \mathbf{S})\}$
E. $\{\langle A, C\rangle \mid \exists B(\langle A, B\rangle \in \mathbf{R} \rightarrow \langle B, C\rangle \in \mathbf{S})\}$

---

l. Consider the schema $\mathbf{R}(\underline{\mathsf{A}}, \underline{\mathsf{B}})$, and $\mathbf{S}(\underline{\mathsf{B}}, \underline{\mathsf{C}})$.

A.
```
select distinct R.A, S.C from R, S where R.B = S.B;
```
B.
```
select distinct R.A, S.C from R, S s1
       where R.B in (select s2.B from S s2 where s1.C = s2.C);
```
C. $\pi_{\mathsf{A,C}}(\mathbf{R} \bowtie \mathbf{S})$
D. $\{\langle A, C\rangle \mid \neg \forall B(\langle A, B\rangle \in \mathbf{R} \rightarrow \langle B, C\rangle \notin \mathbf{S})\}$
E. $\{\langle A, C\rangle \mid \exists B(\langle A, B\rangle \in \mathbf{R} \rightarrow \langle B, C\rangle \in \mathbf{S})\}$

---

[1] Questions 3j–3l were botched on the test! The typesetting got messed up. Apologies. Everyone got credit for the three.

They are corrected here. And yes, Question 3k & 3l was so nice I asked it twice! (Also, not intended.)

R

A	B
1	a
2	b
2	c
3	d

S

B	C
b	5
b	6
c	5
d	6
e	5

Two tables: **R** & **S**.

A	B	C
1	b	5
1	b	6
1	c	5
1	d	6
1	e	5
2	b	5
2	b	6
2	c	5
2	d	6
2	e	5
3	b	5
3	b	6
3	c	5
3	d	6
3	e	5

I

A	B	C
1	a	5
1	a	6
2	b	5
2	b	6
2	c	5
2	c	6
3	d	5
3	d	6

II

A	B	C
1	a	5
2	b	6
2	c	5
3	d	6

III

A	B	C
2	b	5
2	b	6
2	c	5
3	d	6

IV

A	B	C

V

Possible answer tables.

For Questions 3m—3o, consider the tables above. All the joins below are natural joins.

---

m. What is the resulting table of $\{\langle A, B, C\rangle \mid \langle A, B\rangle \in \mathbf{R} \wedge \langle B, C\rangle \in \mathbf{S}\}$?

A. I **B. II** **C. III** **D. IV** **E. V**

---

n. What is the resulting table of $\{\langle A, B, C\rangle \mid \langle A, B\rangle \in \mathbf{R} \wedge \exists\, B(\langle B, C\rangle \in \mathbf{S})\}$?

A. I **B. II** **C. III** **D. IV** **E. V**

---

o. What is the resulting table of $\{\langle A, B, C\rangle \mid \langle A, B\rangle \in \mathbf{R} \wedge \langle B, A\rangle \in \mathbf{S}\}$?

A. I **B. II** **C. III** **D. IV** **E. V**

4. (10 points) **Aggregation.** *The sum is greater...* [Exercise]

   Consider the schema in Figure 1 in Question 2 again.

   a. (5 points) Write an SQL query that lists actors (**Actor**) by p# and name with the number of movies (#movies) in which they have been cast (**Cast**) for which they were *also* a *writer* (**Writer**) of the movie's screenplay (**ScreenPlay**) *or* for which they were the movie's *director* (**Director**).[2]

```
with
    InMovie (title, studio, year, actor) as (
        select distinct title, studio, year, actor
        from Cast
    )
select P.p#, P.name, count(*) as #movies
from InMovie C, Person P
where C.actor = P.p#
  and ( (C.actor = (
                select M.director
                from Movie M
                where  C.title = M.title
                  and   C.year = M.year
                  and C.studio = M.studio
            )
        ) or
        (C.actor in (
                select A.writer
                from Authored A
                where C.title = A.title
                  and  C.year = A.year
            )
        )
      )
group by P.p#, P.name;
```

[2] Any actor who has appeared in *no* such movies does not have to be reported by the query.

b. (5 points) Write an SQL query that reports any actor by p#, name, and nationality and movie by title, studio, year, and genre *if* the actor acted in that movie (**Cast**) *and* the time the actor appears in the movie (the *sum* of the *minutes* over the actor's *roles*) *exceeds* the length the movie (**Movie**.length, which is in minutes).

```
select P.p#, P.name, P.nationality,
       M.title, M.studio, M.year, M.genre
from Cast C, Person P, Movie M
where  C.actor = P.p#
  and  C.title = M.title
  and   C.year = M.year
  and C.studio = M.studio
group by P.p#, P.name, P.nationality,
         M.title, M.studio, M.year, M.genre
having sum(C.minutes) > M.length;
```

Extra space.

Extra space.

Extra space.

Relax. Turn in your exam. Go home.