EECS-3421A: Test #2

"Queries"

Electrical Engineering & Computer Science
Lassonde School of Engineering
York University

Family Name: ______________________________
Given Name: ______________________________
Student#: ______________________________
EECS Account: ______________________________

Instructor: Parke Godfrey
Exam Duration: 75 minutes
Term: Fall 2016

Instructions

- **rules**
  - The test is closed-note, closed-book. Use of a calculator is permitted.
- **answers**
  - Should you feel a question needs an assumption to be able to answer it, write the assumptions you need along with your answer.
  - If you need more room to write an answer, indicate where you are continuing the answer.
  - For multiple choice questions, choose *one* best answer for each of the following. There is no negative penalty for a wrong answer.
- **notation**
  - For schema, the underlined attributes indicate a table's primary key (and are, hence, not nullable). Attributes in *italics* are not nullable. Foreign keys are indicated by FK.
  - Assume *set* semantics for relational-algebra expressions.
- **points**
  - The number of points a given question is worth is marked.
  - There are five major parts worth 10 points each, for 50 points in total.

Marking Box	
1.	/10
2.	/10
3.	/10
4.	/10
5.	/10
Total	/50

1. [10pt] **Relational Algebra.** *Quadratic queries!* [SHORT ANSWER]

   For Questions 1a to 1c, use the Colours schema in Figure 1 on page 15 (as used in examples in class) to write *relational-algebra* queries for the English questions posed.

---

a. [2pt] Show customers by cust# and cname who have bought a *pink* (colour) *Lamborghini* (prod#).

$$\pi_{cust\#,cname}\left((\textbf{Customer} \bowtie \sigma_{colour='pink'}(\textbf{Item})) \bowtie \sigma_{pname='Lamborghini'}(\textbf{Product})\right)$$

marking guide
+1 correct ⋈'s and sources
+1 correct σ's
Gave credit for "$\sigma_{prod\#='Lamborghini'}$," given the wording.

---

b. [2pt] Show products by prod# and pname that are owned by at least two customers.

$$\pi_{prod\#,pname}\left(\textbf{Product} \bowtie (\pi_{cust\#\neq cust\#2}\left(\textbf{Item} \bowtie \pi_{cust\#\rightarrow cust\#2,prod\#,pname}(\textbf{Item})\right))\right)$$

marking guide
+1 correct sources & π's
+1 proper self-join on Item

---

c. [2pt] For each available colour (colour), show the most expensive product by prod# and pname that comes in that colour. (In case of ties for most expensive, list all in the tie.)

$$
\begin{aligned}
&\pi_{colour}(\pi_{colour,prod\#,name}(\textbf{Product} \bowtie \textbf{Avail\_colour})\\
&\qquad -\\
&\qquad \pi_{colour,prod\#,name}(\sigma_{cost>cost2}(\\
&\qquad\qquad \pi_{colour,prod\#,name,cost}(\textbf{Product} \bowtie \textbf{Avail\_colour})\\
&\qquad\qquad \bowtie\\
&\qquad\qquad \pi_{colour,prod\#,name,cost\rightarrow cost2}(\textbf{Product} \bowtie \textbf{Avail\_colour})\\
&\qquad))\\
&)
\end{aligned}
$$

marking guide
+1 correct structure
+1 correct logic to eliminate not most expensive

R

A	B
a	b
c	b
e	f
g	h
i	h

S

B	C
f	b
b	d
h	a
h	c
b	e

T

A	C
g	a
a	b
i	c
e	d
c	e

Consider the three tables **R**, **S**, & **T** above for Questions 1d & 1e.

---

d. [2pt] Show the results of **R** $\bowtie$ **S**.

A	*B*	*C*
a	*b*	*d*
a	*b*	*e*
c	*b*	*d*
c	*b*	*e*
e	*f*	*b*
g	*h*	*a*
g	*h*	*c*
i	*h*	*a*
i	*h*	*c*

marking guide

1 if extra tuple *or* missing tuple

0 if more wrong than that

---

e. [2pt] Show the results of **R** $\bowtie$ (**S** $\bowtie$ **T**).

A	*B*	*C*
c	*b*	*e*
g	*h*	*a*
i	*h*	*c*

marking guide

1 if extra tuple *or* missing tuple

0 if more wrong than that

2. (10 points) **SQL.** *Some Quidditch League!* [Exercise]

Consider the Movie database with the schema in Figure 2 on page 15 for the questions below.

---

a. [5pt] Write an `SQL` query for the following.

List each actor by name, gender, role, and the minutes they appear on screen in that role in a given movie by title, studio, and year such that "Hampton Fancher" was an author of the screenplay of the movie and the movie's genre is "SciFi".

```
select P.name, P.gender, C.role, C.minutes, M.title, M.studio, M.year
from Person P, Cast C, Movie M, Authored A, Person W
where C.actor = P.p#
  and C.title = M.title and C.studio = M.studio and C.year = M.year
  and M.genre = 'SciFi'
  and A.title = M.title and A.year = M.year
  and A.writer = W.p#
  and W.name = 'Hampton Fancher';
```

marking guide

+2 correct sources and schema

+2 $\bowtie$'s correct

+1 σ's correct

- Note that 'Hampton Fancher' is a Person's *name*, *and* not the value of `p#` in Writer.

b. [5pt] For each movie, report how many male actors and how many female actresses—gender = 'M' for *male* and gender = 'F' for *female*—were *cast* in the movie in columns male and female, respectively.

Only count a given actor or actress *once* per movie; that is, that a person may have played several *characters* (*roles*) in the movie does *not* count multiple times.

For full credit, the column male or female should report '0' (zero) if there were no actors or actresses, respectively, in the movie.

```
with
    Counts (title, studio, year, male, female) as (
        select title, studio, year, 0, 0
            from Movie
        union
        select C.title, C.studio, C.year, count(distinct C.actor), 0
            from Cast C, Person P
            where C.actor = P.p#
              and P.gender = 'M'
        union
        select C.title, C.studio, C.year, 0, count(distinct C.actor)
            from Cast C, Person P
            where C.actor = P.p#
              and P.gender = 'F'
    )
select C.title, C.studio, C.year,
       max(C.male) as male,
       max(C.female) as female
    from Counts C
    group by C.title, C.studio, C.year;
```

marking guide

+2 counts per gender are done correctly
+1 accounts for the 0's
+1 right logic to get the correct numbers
+1 correct schema, ⋈'s, structure of query

3. (10 points) **Query Logic.** *Take the **L** train to **Q** Street.* [Analysis]

---

a. [2pt] State in *plain, concise English* what the following SQL query over the *Movie database* does. (See the Movie schema in Figure 2 on page 15.)

Note that you will 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!)

```
select P.p#, P.name as actor, D.p# as d#, D.name as director
from Person P, Person D
where not exists (
            select *
            from Movie M
            where M.director = D.p#
              and not exists (
                    select *
                    from Cast C
                    where C.actor = P.p#
                       and C.title = M.title
                       and C.studio = M.studio
                       and C.year = M.year
                )
        )
   and exists (
            select *
            from Movie M
            where M.director = D.p#
        );
```

What actor has appeared in all the movies directed by a director, such that director has directed some movie? List all such pairs of actors and directors.

marking guide

0: not close; 1: caught some of the logic; 2: correct.

---

b. [2pt] Is the *having* clause *logically redundant* in SQL? That is, could one always write the "same" query not using the *having* clause?

Explain briefly why or why not.

It is redundant *in that we can always write the "same" query w/o use of the* having *clause. This can be done by making our query with the* group by *a sub-query, then using the* where *clause in the main query to filter the "aggregated" tuples.*

marking guide

2 stating it is redundant, and giving a suitable explanation

1 stating it is redundant, but explanation lacking *or* stating it is not, but explanation meaningful

0 no proper explanation

c. [2pt] Given $\mathbf{R}(\underline{A}, \underline{B})$ and $\mathbf{S}(\underline{B}, \underline{C})$, rewrite

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

as an equivalent SQL query.

```
select distinct Z.A, X.C
from R Z, S X
where Z.B not in (
        select Y.B
        from S Y
        where X.C = Y.C);
```

---

d. [2pt] Consider relations $\mathbf{R}(\underline{A}, \underline{B})$, $\mathbf{S}(\underline{B}, \underline{C})$, and $\mathbf{T}(\underline{C}, \underline{D})$.
Can $((\mathbf{R} \bowtie \mathbf{S}) \bowtie \mathbf{T})$ and $((\mathbf{R} \bowtie \mathbf{T}) \bowtie \mathbf{S})$ evaluate to different answer sets, or must they evaluate to the same answer set?
Show how they could evaluate differently, *or argue* why they must evaluate to the same.

The join operator is both associative *and* commutative. *Therefore,* $(\mathbf{R} \bowtie \mathbf{S}) \bowtie \mathbf{T} = \mathbf{R} \bowtie (\mathbf{S} \bowtie \mathbf{T}) = \mathbf{R} \bowtie (\mathbf{T} \bowtie \mathbf{S}) = (\mathbf{R} \bowtie \mathbf{T}) \bowtie \mathbf{S}$.
Therefore, they must evaluate to the same.

---

e. [2pt] Consider the relations $\mathbf{R}(\underline{A}, B)$ and $\mathbf{S}(\underline{C}, D)$, and query

```
select distinct A, B from R
where exists (select * from S where A = C);
```

Write this as a relational-algebra expression.

$$\pi_{A,B}(\mathbf{R} \bowtie_{A=C} \mathbf{S})$$

4. [10pt] **Normalization.** *It's the end of the world as we know it...* [Analysis]

a. [3pt] Consider the relation **T** with attributes A, B, C, D, and E, and with the following functional dependencies (FDs):

$$AB \mapsto C \qquad C \mapsto A$$
$$AD \mapsto E \qquad C \mapsto D$$

Dr. Datta Bas claims that the decomposition step of **T** into ABC & CDE is a *lossless-join* decomposition step.

Explain convincingly *either* that he is correct *or* that he is wrong.

$\{C\} = \{A, B, C\} \cap \{C, D, E\}$. *So, we ask whether* $C \mapsto AB$ *or* $C \mapsto DE$. $C* \mapsto ADE$, *so* $C \mapsto DE$*!*

This means Dr. Bas is right! The two parts can be joined back losslessly.

marking guide

+2 "Yes", plus proper reasoning: identifies intersection of common attr's is key for one, and shows by reasoning over FDs

+1 "Yes", but explanation lacking

b. [2pt] Consider the relation **R** with attributes A, B, and C, and with just the one following functional dependency (FD):

$$A \mapsto BC$$

Consider the decomposition of **R** into AB and BC. Construct a counterexample (example tuples) that demonstrate that this decomposition is *not* lossless (that is, it is *lossy*).

A	B	C
1	2	3
4	2	5

⇒

A	B
1	2
4	2

&

B	C
2	3
2	5

$AB \bowtie BC$ =

A	B	C
1	2	3
1	2	5
4	2	3
4	2	5

∴ *It is lossy.*

marking guide

2 must have given counterexample, as requested

1 explained convincingly, but no counterexample

0 did not explain convincingly

For Questions 4c to 4e, consider the relation **R** with attributes A, B, C, and D, and with the following functional dependencies (FDs):

$$AB \mapsto C \qquad A \mapsto D$$
$$BD \mapsto C$$

---

c. [1pt] What are the keys of **R**?

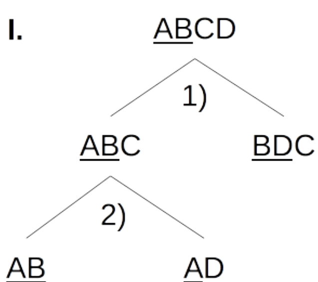
> *AB is the only key of* **R**.

---

d. [3pt] Decompose **R** losslessly into a BCNF decomposition.

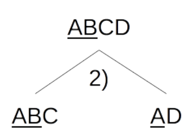
> *1)* $BD \mapsto C$ *: violates BCNF*
>
> *2)* $A \mapsto D$ *: violates 2NF*

I.



or

II.



Note that decomposing on $AB \mapsto C$ *is useless, as that FD does* not *violate BCNF for the relation.*

---

e. [1pt] Is your decomposition in your answer to Question 4d dependency preserving? Why or why not?

> ***I.*** *No, it is not. The attr's of* $AB \mapsto C$ *do not appear together in any of the final tables.*
> ***II.*** *No, it is not. The attr's of* $BD \mapsto C$ *do not appear together in any of the final tables.*

5. [10pt] **General.** *... and I feel fine.* [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 that you feel a clarification to your answer is needed, write a brief clarification on the side.

Let $|\mathbf{T}|$ denote the number of tuples in $\mathbf{T}$.

---

a. [1pt] Consider the schema

$\mathbf{R}(\underline{A}, B)$ FK (B) refs $\mathbf{S}$

$\mathbf{S}(A, \underline{B})$ FK (A) refs $\mathbf{R}$

Note that none of the attributes are nullable. Which of the following is guaranteed to produce as many as, or more, tuples than each of the others?

A. $\mathbf{R} \bowtie \mathbf{S}$
B. $\pi_{A}(\mathbf{R}) \bowtie \mathbf{S}$
[C.] $\pi_{A}(\mathbf{R}) \bowtie \pi_{B}(\mathbf{S})$
D. $\mathbf{R} \bowtie \pi_{B}(\mathbf{S})$
E. *There is not enough information to answer this.*

---

b. [1pt] Assume $|\mathbf{R}| > 0$ and $|\mathbf{S}| > 0$. If one natural joins tables $\mathbf{R}$ and $\mathbf{S}$, but $\mathbf{R}$ and $\mathbf{S}$ have no column names in common, then

A. it is an *error*.
B. the answer set is an empty table.
C. it is an outer join.
D. it is the same as $\mathbf{R} \cap \mathbf{S}$.
[E.] it is the same as $\mathbf{R} \times \mathbf{S}$.

For Questions 5c & 5d, consider the schema

$\mathbf{R}(\underline{A}, B)$ FK (B) refs $\mathbf{R}$ (A)

---

c. [1pt] What is the *smallest* that $|\mathbf{R} \bowtie \pi_{A \to B, B \to A}(\mathbf{R})|$ can be?

[A.] 0
B. $|\mathbf{R}|$
C. $\frac{1}{2}|\mathbf{R}|$
D. $2|\mathbf{R}|$
E. $|\mathbf{R}|^2$

---

d. [1pt] What is the *largest* that $|\mathbf{R} \bowtie \pi_{A \to B, B \to A}(\mathbf{R})|$ can be?

A. 0
[B.] $|\mathbf{R}|$
C. $\frac{1}{2}|\mathbf{R}|$
D. $2|\mathbf{R}|$
E. $|\mathbf{R}|^2$

---

e. [1pt] The technique of *synthesis* for normalization always

A. works only if there will be no multi-attribute keys.
B. achieves a 3NF schema, but it may not be dependency-preserving.
[C.] achieves a dependency-preserving, 3NF schema.
D. achieves a BCNF schema, but it may not be dependency-preserving.
E. achieves a dependency-preserving, BCNF schema.

---

f. [1pt] Which of the following `SQL` queries is illegal?

A. `select * from T;`
B. `select count(*) from T;`
C. `select count(*) from T group by A;`
D. `select count(*), max (B) from T group by A;`
[E.] `select max(count(*)) from T group by A;`

---

g. [1pt] Which of the following `SQL` queries is illegal?

A. `select A from T;`
[B.] `select A, count(*) from T;`
C. `select A, count(*) from T group by A;`
D. `select A, count(*) from T group by A, B;`
E. `select A, B, count(*) from T group by A, B;`

---

h. [1pt] Consider the schema $\mathbf{R}(\underline{A}, B)$, $\mathbf{S}(\underline{A}, \underline{D})$, and $\mathbf{T}(\underline{D}, B)$. One of the following relational-algebra expressions is not like the others. That is, one of them may evaluate differently from the other four. Which one?

A. $\pi_B(\mathbf{R} \bowtie \mathbf{T} \bowtie \mathbf{S})$
B. $\pi_B((\mathbf{R} \bowtie \mathbf{T}) \cap (\mathbf{S} \times \pi_B(\mathbf{T})))$
[C.] $\pi_B(\mathbf{R} \bowtie \mathbf{S}) \cap \pi_B(\mathbf{S} \bowtie \mathbf{T})$
D. $\pi_B(\mathbf{R} \bowtie \mathbf{S} \bowtie \mathbf{S} \bowtie \mathbf{T})$
E. $\pi_B(\mathbf{R} \bowtie \mathbf{S} \bowtie \mathbf{T})$

---

i. [1pt] Consider table **R**(<u>A</u>, B) for which B is of type *integer* and $|\mathbf{R}| = n > 0$. How many tuples will the query

```
select A from R where B <= 5 or B > 5;
```

return?

A. 0
B. $\frac{1}{2}n$
C. n
D. n^2
[E.] *There is not enough information to answer this.*

---

j. [1pt] Consider the relation **Enrol** with attributes sid, cid, term, and grade which stores academic records of students. Attribute sid is a student identifier and cid is a class—a given section of a course in a given term—identifier.

Here is a query involving **Enrol**:

```
select distinct cid
    from ( select * from Enrol E1
              where not exists
                  (select *
                      from Enrol E2
                      where E2.cid = E1.cid
                          and E2.grade > E1.grade)
          ) as V
    where grade = 7;
```

Which of the following queries must return the same result as the query above?

I.
```
select distinct E.cid
    from Enrol E
    where E1.grade = 7
except
select distinct E.cid
    from Enrol E
    where E1.grade > 7
```

II.
```
select distinct cid
    from Enrol
    group by cid
    having max(grade) = 7;
```

A. **I** only.
[B.] **II** only.
[C.] Both **I** and **II**.
D. Neither **I** nor **II**.
E. There is not enough information available to determine this.

> ***I** has a syntax error due to 'E1' instead of 'E' as intended. So, **B** if **I** errors out; or **C** if reading **I** as correct.*

Extra Space

Extra Space

Relax. Turn in your test. Return to the wild!

Reference *(Detach this page for convenience, if you want.)*

Schema for the Colours Database.

Customer	
cust#	PK
cname	
fav_colour	
phone#	

Item	
item#	PK
prod#	FK to **Product**
cust#	FK to **Customer**
colour	
date_sold	

Product	
prod#	PK
pname	
cost	
maker	FK to **Company**

Avail_Colours	
prod#	PK, FK to **Product**
colour	PK

Figure 1: Colours Schema.

Schema for the Movie Database.

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 2: Movie Schema.

REFERENCE

The Normal Form Definitions.

1NF: Domain of each attribute is an *elementary* type; that is, not a *set* or a *record structure*.

2NF: Whenever $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation **R** and $A \notin \mathcal{X}$, then either

- A is *prime*, or
- $\mathcal{X}$ is not a proper subset of any key for **R**.

3NF: Whenever $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation **R** and $A \notin \mathcal{X}$, then either

- A is *prime*, or
- $\mathcal{X}$ is a key or a super-key for **R**.

BCNF: Whenever $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation **R** and $A \notin \mathcal{X}$, then

- $\mathcal{X}$ is a key or a super-key for **R**.

An attribute A is called *prime* if A is in any of the candidate keys.

Figure 3: The Normal Forms.