CSE 344 Midterm

Friday, Feb. 15, 2019, 1:30-2:20

Name: ______________________________

Question	Points	Score
1	30	
2	25	
3	25	
4	10	
5	10	
Total:	100	

- This exam is CLOSED book and CLOSED devices.
- You are allowed ONE letter-size page with notes (both sides).
- You have 50 minutes;
- Answer the easy questions before you spend too much time on the more difficult ones.
- If there is a line to write your answer (e.g True/False, a/b/c/d, etc), you must write your answer. Circling the option is not enough.
- Good luck!

1 SQL

1. (30 points)

   An online shopping company stores a database about its products, customers, and orders, with the following schema:

   ```
   Product(pid,pname,price)
   Orders(oid,pid,cid,qnt,date)
   Customer(cid,cname,city)
   ```

   Producs have an ID, a name, and a sell price

   Customers have an ID, a name, and the city where they live.

   Each order is for one product and one customer. The attribute `qnt` is the quantity, i.e. the number of units of the product placed in that order; `qnt` > 0.

   (a) (5 points) Write the sequence of SQL statements necessary to create the tables above. The attributes pid, oid, cid, qnt are integers, price is a real number, date is of type DATE and all other attributes are text. Include all keys or foreign keys declarations.

   Solution:

   ```
   DROP TABLE IF EXISTS Orders;
   DROP TABLE IF EXISTS Product;
   DROP TABLE IF EXISTS Customer;

   CREATE TABLE Product(pid INT PRIMARY KEY, pname TEXT, price FLOAT);
   CREATE TABLE Customer(cid INT PRIMARY KEY, cname TEXT, city TEXT);
   CREATE TABLE Orders(oid INT PRIMARY KEY,
                       pid INT REFERENCES Product,
                       cid INT references Customer, qnt INT, date DATE);
   ```

   We graded the key/foreign keys strictly: each was 1 point

   (Note: no need to write `DROP TABLE`. This is only for the instructor to tests this code in SQL.)

```
Product(pid,pname,price)
Orders(oid,pid,cid,qnt,date)
Customer(cid,cname,city)
```

(b) (5 points) Write a SQL query that returns, for each product, the quantity of those products sold to customers in Seattle. Your query should return the pid, product name, and total quantity sold in Seattle, sorted in decreasing order of the total quantity. Your query should include products with a total = 0.

Solution:

```
select x.pid, x.pname, count(y.oid)
from Product x
     left outer join Orders y on x.pid=y.pid
     left outer join Customer z on y.cid=z.cid and z.city='Seattle'
group by x.pid, x.pname
order by count(y.oid);
```

-2 points for missing outer joins
-2-3 points for missing group by
-1 point for other minor mistakes

```
Product(pid,pname,price)
Orders(oid,pid,cid,qnt,date)
Customer(cid,cname,city)
```

(c) (10 points) For each city, return the most expensive product sold in that city. Your query should return the city name, the `pid`, `pname`, and `price` of the most expensive product sold there; in case of a tie (two or more products had the same maximum price), then return all of those products. If nothing was ordered in a city, then your query does not need to return that city.

Solution: Solution 1:

```
with temp as
   (select z.city, max(x.price) as mp
    from Product x, Orders y, Customer z
    where x.pid = y.pid and y.cid = z.cid
    group by z.city)
select u.city, x.price
from Product x, Orders y, Customer z, temp u
where x.pid = y.pid and y.cid = z.cid
  and z.city=u.city and x.price=u.mp;
```

Solution 2:

```
select u.city, w.price
from Product x, Orders y, Customer z,
     Product w, Orders v, Customer u
where x.pid = y.pid and y.cid = z.cid
  and w.pid = v.pid and v.cid = u.cid
  and z.city = u.city
group by u.city, w.price
having w.price >= max(x.price);
```

-2 points for missing `z.city` = `u.city`
-1 point for various smaller mistakes
-5 points for a query that only computes max.

```
Product(pid,pname,price)
Orders(oid,pid,cid,qnt,date)
Customer(cid,cname,city)
```

(d) Consider the following query:

```
--  Q:
select distinct z.cid, z.cname
from Product x, Orders y, Customer z
where x.pid = y.pid and y.cid = z.cid
  and x.price < 200
  and y.qnt > 10
  and z.city = 'Seattle';
```

For each query below, indicate whether return the same answer or not. You only need to answer Y or N.

i. (2 points) Is this query equivalent to Q?

```
--  Q1:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200
  and y2.qnt > 10
  and z.city = 'Seattle';
```

i. **N**

Yes or No?

ii. (2 points) Is this query equivalent to Q?

```
--  Q2:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200
  and y1.qnt > 10
  and z.city = 'Seattle';
```

ii. **Y**

Yes or No?

iii. (2 points) Is this query equivalent to Q?

```
-- Q3:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200  and x2.price < 300
  and y1.qnt > 10 and y2.qnt > 5
  and z.city = 'Seattle';
```

iii. **Y**

Yes or No?

iv. (2 points) Is this query equivalent to Q?

```
-- Q4:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200  and x2.price < 100
  and y1.qnt > 10 and y2.qnt > 50
  and z.city = 'Seattle';
```

iv. **N**

Yes or No?

v. (2 points) Is this query equivalent to Q?

```
-- Q5:
select distinct z.cid, z.cname
from Product x1, Product x2, Orders y1, Orders y2, Customer z, Customer z2
where x1.pid = y1.pid and y1.cid = z.cid
  and x2.pid = y2.pid and y2.cid = z.cid
  and x1.price < 200
  and y1.qnt > 10
  and z.city = 'Seattle' and z2.city = 'Portland';
```

v. **N**

Yes or No?

2 Relational Algebra

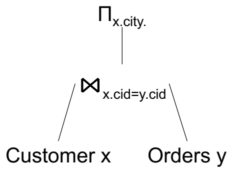
2. (25 points)

Consider the same relational schema as before:

```
Product(pid,pname,price)
Orders(oid,pid,cid,qnt,date)
Customer(cid,cname,city)
```

(a) (5 points) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

Hint: to avoid renaming, use aliases in the query plan, like this

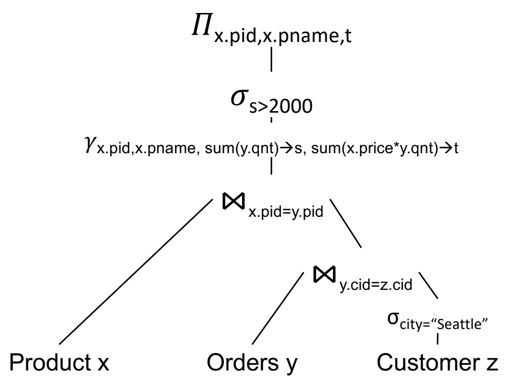


```
select x.pid, x.pname, sum(x.price*y.qnt) as revenue
from Product x, Orders y, Customer z
where x.pid = y.pid and y.cid = z.cid
  and z.city = 'Seattle'
group by x.pid, x.pname
having sum(y.qnt) > 2000;
```

Write the Relational Query expression below:

Solution:

$$\Pi_{\texttt{x.pid,x.pname},t}\left(\sigma_{s>2000}\left(\gamma_{\texttt{x.pid,x.pname,sum(y.qunt)}\to s,\texttt{sum(x.price*y.qnt)}\to t}\left(\texttt{Product x} \ltimes_{\texttt{x.pid=y.pid}}\left(\texttt{Orders y} \ltimes_{\texttt{y.cid=z.cid}} \sigma_{\texttt{city="Seattle"}}\texttt{Customer z}\right)\right)\right)\right)$$



```
Product(pid,pname,price)
Orders(oid,pid,cid,qnt,date)
Customer(cid,cname,city)
```

(b) i. (2 points) Which of the following is the most accurate English interpretation of the SQL query below?

```
select distinct z.city
from Customer z
where not exists
       (select *
        from Orders y
        where y.cid = z.cid and y.qnt < 100);
```

Returns cities that ...

(A) had at least one order for <100 units of a product.

(B) have a customer who placed at least one order for more <100 units of a product.

(C) had only orders with ≥ 100 units.

(D) have a customer that placed only orders with ≥ 100 units.

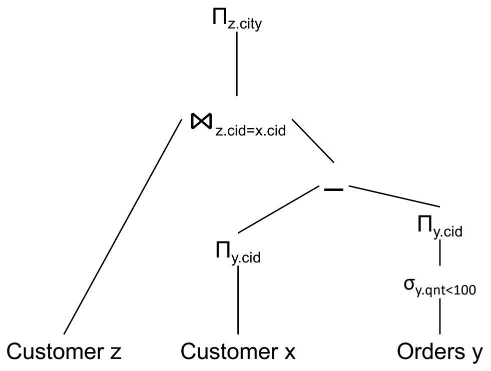
i. **D**

A/B/C/D:

ii. (8 points) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query above. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

Solution:

$$\Pi_{\texttt{z.city}}(\texttt{Customer z} \bowtie_{\texttt{z.cid=x.cid}} \Pi_{x.cid}(\texttt{Customer x}) - \Pi_{y.cid}(\sigma_{y.qnt<100}(\texttt{Orders y})))$$



-2 The most common mistake by far was doing the subtraction on cities rather than on cid.
-1/-2 Common syntax issues and other small misc RA mistakes.

```
Product(pid,pname,price)
Orders(oid,pid,cid,qnt,date)
Customer(cid,cname,city)
```

(c) We continue to use the same schema as before; remember that some attributes are keys, and others are foreign keys. We further assume that the database has no NULLs. Answer the questions below, assuming all relational algebra expressions have set semantics. The notation $|S|$ means the cardinality of a set S (number of tuples in S).

i. (2 points) Does the following always hold?

$$|(\texttt{Product x}) \bowtie_{\texttt{x.pid=y.pid}} \sigma_{\texttt{y.qnt}>20}(\texttt{Orders y})| = |\texttt{Orders}|$$

i. **No**

Yes/No:

ii. (2 points) Does the following always hold?

$$|(\texttt{Product x}) \bowtie_{\texttt{x.pid=y.pid}} \sigma_{\texttt{y.qnt}>20}(\texttt{Orders y})| = |\sigma_{\texttt{qnt}>20}(\texttt{Orders})|$$

ii. **Yes**

Yes/No:

iii. (2 points) Does the following always hold?

$$\sigma_{\texttt{y.qnt}>20}(\texttt{Orders y}) \bowtie_{\texttt{x.pid=y.pid}} (\texttt{Product x}) =$$
$$= \sigma_{\texttt{y.qnt}>20}(\texttt{Product x} \bowtie_{\texttt{x.pid=y.pid}} (\texttt{Orders y})$$

iii. **Yes**

Yes/No:

iv. (2 points) Does the following always hold?

$$\gamma_{\texttt{x.city,count(*)}\to\texttt{cnt}}((\texttt{Customer x}) \bowtie_{\texttt{x.cid=y.cid}} (\texttt{Orders y})) =$$
$$= \Pi_{\texttt{x.city,cnt}}((\texttt{Customer x}) \bowtie_{\texttt{x.cid=y.cid}} \gamma_{\texttt{y.cid,count(*)}\to\texttt{cnt}}(\texttt{Orders y}))$$

iv. **No**

Yes/No:

v. (2 points) Does the following always hold?

$$\gamma_{\texttt{x.city,count(*)}\to\texttt{cnt}}((\texttt{Customer x}) \bowtie_{\texttt{x.cid=y.cid}} (\texttt{Orders y})) =$$
$$= \gamma_{\texttt{x.city,sum(cnt)}\to\texttt{cnt}}((\texttt{Customer x}) \bowtie_{\texttt{x.cid=y.cid}} \gamma_{\texttt{y.cid,count(*)}\to\texttt{cnt}}(\texttt{Orders y}))$$

v. **Yes**

Yes/No:

3 Datalog

3. (25 points)

Consider a database storing the employees of a company:

`Employee(eid,name,office,mid)`

The attributes `eid`, `name`, and `office` are the Employee's ID, name, and office, while `mid` is the Manager's ID. The CEO of the company is called `‘‘Smith’’`. Every employee has a manager, except for the CEO, who is managing himself (`eid` = `mid`). An *indirect manager* of an employee is either her manager, or her manager's manager, etc, up to the CEO.

Answer the questions below.

(a) (5 points) Write a datalog program that returns the Employee ID's and names of all employees managed directly or indirectly by `‘‘Alice’’`.

Solution:

Compute the ID and name at the same time:

```
Q(e,n) :- Empolyee(e,n,_,m), Employee(m,"Alice",_,_)
Q(e,n) :- Employee(e,n,_,m), Q(m,_)
```

Better: compute only the ID's, later fetch the names:

```
M(e) :- Employee(e,_,_,m), Employee(m,"Alice",_,_)
M(e) :- Empolyee(e,_,_,m), M(m)
Q(e,n) :- M(e), Employee(e,n,_,_)
```

1 point for base case
3 point for recursive case (Partial credit for the attempt/logic)
1 point for returning the correct values/general Datalog Syntax

(b) (10 points) The company's requirement is that every employee should be managed directly or indirectly by the CEO, `‘‘Smith’’`. However, the database got corrupted and this requirement became violated. Write a datalog program to find all employees that are not managed directly or indirectly by `‘‘Smith’’`. Your program should return their Employee ID and name.

Solution: First compute `M(e)` above, then:

```
Q(e,n) :- Employee(e,n,_,_),!M(e)
```

most common mistakes:
missing recursion (-5)
using 'smith' as an id (-2)

(c) (10 points) Two managers are called *relatives* if they manage (directly) two employees sharing the same office, or if they are relatives of some other common manager (i.e. relatives by transitivity). For example if `Alice` and `Bob` share the same office, then their direct managers `Carol` and `David` are relatives; furthermore, if `David` is also a relative of `Eve`, then `Carol` and `Eve` are relatives by transitivity. Write a datalog program to find all relatives of `Carol`. Your program should return their Employee ID's and names.

Solution:

```
R(e1,e2) :- Employee(m1,_,o,e1),Employee(m2,_,o,e2)
R(e1,e3) :- R(e1,e2),R(e2,e3)
Q(e,n) :- Empolyee(ec,"Carol",_,_), R(ec,e),Empolyee(e,n,_,_)
```

If students took the approach of starting with Carol in the base case, then -5 if they did not include the recursive case.

If students took the approach of computing all relatives, then -3 if they forgot to restrict the relatives to Carol and -3 if they forgot the recursive case.

-1 if they used "Carol" as an employee id.

4 JSON and SQL++

4. (10 points)

(a) Answer the following questions. In your answer you may omit apostrophes, i.e. you may write {A:a1} instead of {'A':'a1'} .

i. (1 point) What does the following SQL++ query return?

```
with t as
 [{'A':'a1', 'F':[{'B':'1'}, {'B':'2'}], 'G':[{'C':'1'},{'C':'2'}]},
  {'A':'a2', 'F':[{'B':'3'}, {'B':'4'},{'B':'5'}], 'G':[ ]},
  {'A':'a3', 'F':[{'B':'2'}], 'G':[{'C':'1'},{'C':'3'}]}]
Select x.A
From  t  x, x.G y
where y.C='1';
```

Solution:

```
{ "A": "a1" }
{ "A": "a3" }
```

ii. (1 point) What does the following SQL++ query return?

```
with t as
 [{'A':'a1', 'F':[{'B':'1'}, {'B':'2'}], 'G':[{'C':'1'},{'C':'2'}]},
  {'A':'a2', 'F':[{'B':'3'}, {'B':'4'}, {'B':'5'}], 'G':[ ]},
  {'A':'a3', 'F':[{'B':'2'}], 'G':[{'C':'1'},{'C':'3'}]}]
Select x.A, y.B
From  t  x, x.F y;
```

Solution:

```
{ "A": "a1", "B": "1" }
{ "A": "a1", "B": "2" }
{ "A": "a2", "B": "3" }
{ "A": "a2", "B": "4" }
{ "A": "a2", "B": "5" }
{ "A": "a3", "B": "2" }
```

iii. (1 point) What does the following SQL++ query return?

```
with t as
 [{'A':'a1', 'F':[{'B':'1'}, {'B':'2'}], 'G':[{'C':'1'},{'C':'2'}]},
  {'A':'a2', 'F':[{'B':'3'}, {'B':'4'}, {'B':'5'}], 'G':[ ]},
  {'A':'a3', 'F':[{'B':'2'}], 'G':[{'C':'1'},{'C':'3'}]}]
Select x.A, y.B
From  t  x, x.F y, x.G z
where y.B=z.C ;
```

Solution:

```
{ "A": "a1", "B": "1" }
{ "A": "a1", "B": "2" }
```

iv. (1 point) What does the following SQL++ query return?

```
with t as
 [{'A':'a1', 'F':[{'B':'1'}, {'B':'2'}], 'G':[{'C':'1'},{'C':'2'}]},
  {'A':'a2', 'F':[{'B':'3'}, {'B':'4'}, {'B':'5'}], 'G':[ ]},
  {'A':'a3', 'F':[{'B':'2'}], 'G':[{'C':'1'},{'C':'3'}]}]
Select x1.A as A1, x2.A as A2, y.B
From  t x1, t x2, x1.F y, x2.G z
where y.B=z.C ;
```

Solution:

```
{ "A1": "a1", "A2": "a1", "B": "1" }
{ "A1": "a1", "A2": "a3", "B": "1" }
{ "A1": "a1", "A2": "a1", "B": "2" }
{ "A1": "a2", "A2": "a3", "B": "3" }
{ "A1": "a3", "A2": "a1", "B": "2" }
```

(b) For each statement below, indicate whether it is true or false.

i. (1 point) An *OLTP workload* means a workload of queries that have many joins, aggregates, and very few or no updates.

True/False:

i. **false**

ii. (1 point) An *OLAP workload* means a workload of queries that have many joins, aggregates, and very few or no updates.

True/False:

ii. **true**

iii. (1 point) *Physical data independence* means that the data is compressed.

True/False:

iii. **false**

iv. (1 point) The key-value pair data model is better suited for complex queries (with aggregates) than for simple queries (single lookups).

True/False:

iv. **false**

v. (1 point) NoSQL systems typically run on a large, distributed cluster (meaning: many servers).

True/False:

v. **true**

vi. (1 point) NoSQL systems support physical data independence better than relational database systems.

True/False:

vi. **false**

5 Miscellaneous

5. (10 points)

   Answer each question below.

   (a) For each statement below, indicate whether it is true or false:

   i. (1 point) The First Normal Form means that an attribute of a relation cannot be another relation.

   i. **True**

   True or false?

   ii. (1 point) In the relational data model the data is in First Normal Form.

   ii. **True**

   True or false?

   iii. (1 point) In JSON the data is in First Normal Form.

   iii. **False**

   True or false?

   iv. (1 point) Consider the relation `Product(productName, manufacturer, price, weight)` where the key is `(productName, manufacturer)`. Can there be two different products with the same value of `manufacturer`?

   iv. **Yes**

   Yes or no?

   v. (1 point) If an attribute in a table is a foreign key, then the table cannot contain two tuples with the same value of that attribute.

   v. **False**

   True or false?

(b) In this and the following question we will assume that the relational algebra has set semantics. The relation $R(A, B)$ has m tuples, and the relation $S(B, C)$ has n tuples. You are asked to indicate the largest possible size of natural join $R \bowtie S$, in each of the cases below. Choose one of the following:

(a) m

(b) n

(c) mn

(d) $m + n$

(e) $\max(m, n)$

(f) $(m + n)/2$

(g) None of the above.

i. (1 point) Assume that B is a key in S, and a foreign key in R. What is the largest possible size of the output of the natural join $R \bowtie S$?

i. **(a)**

Answer a-g:

ii. (1 point) Assume that B is a key in R, and a foreign key in S. What is the largest possible size of the output of the natural join $R \bowtie S$?

ii. **(b)**

Answer a-g:

iii. (1 point) No keys or foreign keys were declared in R or S. What is the largest possible size of the output of the natural join $R \bowtie S$?

iii. **(c)**

Answer a-g:

(c) For each question below, indicate whether it is true or false.

i. (1 point) Consider a datalog program consisting of a single, non-recursive rule. Then the program can be rewritten in relational algebra.

i. **true**

True or false?

ii. (1 point) Consider a datalog program consisting of multiple rules, including recursion. Then the program can be rewritten in relational algebra.

ii. **false**

True or false?