CSE344 Midterm Exam
Fall 2016
November 7, 2016

- Please read all instructions (including these) carefully.
- **This is a closed-book exam. You are allowed a one-page handwritten cheat sheet.**
- Write your name and UW student number below.
- No electronic devices are allowed, including **cell phones** used merely as watches. Silence your cell phones and place them in your bag.
- Solutions will be graded on correctness and ***clarity***. Each problem has a relatively simple and straightforward solution. Partial solutions will be graded for partial credit.
- There are 6 pages in this exam, not including this one.
- There are 3 questions, each with multiple parts. If you get stuck on a question move on and come back to it later.
- You have 50 minutes to work on the exam.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the last blank page as scratch paper. **Do not** use any additional scratch paper. Good luck!

By writing your name below, you certify that you have not received any unpermitted aid for this exam, and that you will not disclose the contents of the exam to anyone in the class who has not taken it.

NAME: _______Solutions___________________

STUDENT NUMBER: ______________________________

Problem	**Points**
1	/ 10
2	/ 52
3	/ 38
Total	/ 100

Problem 1: Warm up (10 points total)

Select either True or False for each of the following questions. For each question you get 1 point for answering it correctly, -0.5 point for an incorrect answer, and 0 point for no answer. The minimum you will get for this entire problem is 0.

a) The arity of the relation R(A int, B int) with tuples (4,2), (2,2), (4,2), (3,3), (4,2) is 3.

True **<u>False</u>**

b) Data is encoded in JSon documents using key-value pairs.

<u>True</u> False

c) A relation can have a clustered index on a set of attributes.

<u>True</u> False

d) A datalog rule is safe if and only if every variable appears in a relational atom.

True **<u>False</u>**

e) A SQL query with `GROUP BY` can select all attributes that are grouped on, joined upon, or are aggregates.

True **<u>False</u>**

f) Every natural join can be rewritten using cross product and selection.

True **<u>False</u>**

g) A foreign key does not have to reference a primary key.

<u>True</u> False

h) An inner join between relations R and S always includes all tuples from R in the result.

True **<u>False</u>**

i) Queries with universal quantifiers cannot be unnested.

<u>True</u> False

j) A subquery in the `SELECT` clause in SQL can refer to tuple variables defined in the `WHERE` clause.

<u>True</u> False

Problem 2: Writing Queries (52 points total)

Write the following queries using the schema below:

```
Product (pid, name, cid)  -- cid is foreign key to Company.cid
Company (cid, cname, city)
Purchase(pid, custId, quantity, price)
-- pid is foreign key to Product.pid, custId is foreign key to Customer.custId
Customer(custId, name, city)
```

a) Write a SQL query that returns the name of companies, along with the number of products sold, for companies that have sold at least 2 different types of products anywhere. (13 points)

```
SELECT c.cname, count(*)
FROM product p, company c, purchase p2
WHERE p.pid = p2.pid and c.cid = p.cid
GROUP BY c.cid, c.name
HAVING COUNT(*) > 2
```

b) Write a relational algebra query that returns the distinct names of all customers from Seattle who purchased any one type of product with quantity > 10. Write your query as a tree or a single relational algebra expression. (13 points)

$\delta(\pi_{name}(\sigma_{city=\text{"Seattle"}}(\text{customer}) \bowtie_{custId=custId} \sigma_{quantity>10}(\text{purchase})))$

Schema repeated here for your reference:

```
Product (pid, name, cid)  -- cid is foreign key to Company.cid
Company (cid, cname, city)
Purchase(pid, custId, quantity, price)
-- pid is foreign key to Product.pid, custId is foreign key to Customer.custId
Customer(custId, name, city)
```

c) Write a domain-independent relational calculus query that returns the CIDs of all companies where all of their products have been sold at least once. (13 points)

```
A(x) = ∃y ∃z Company(x,y,z) ∧ ∀a(∃b Product(a,b,x) ⇒ ∃d ∃e ∃f
Purchase(a, d, e, f))
```

d) Write a safe Datalog+negation program that returns the PIDs of the products that have never been sold, along with the names of the companies that made them. Label your answer relation `Ans`. (13 points)

```
NonAnswers(n, p) :- Product(p, _, c), Company(c, n, _),
                    Purchase(p, i, _, _), Customer(i, _, _)
All(n,p) :- Product(p, _, c), Company(c, n, _)

Ans(n,p) :- All(n,p), not NonAnswers(n,p)
```

Problem 3: Short Questions (38 points total)

For the following schema:

```
Purchase(pid, custId, quantity, price)
-- pid is foreign key to Product.pid, custId is foreign key to Customer.custId
Customer(custId, name, city)
```

With statistics:

T(Purchase) = 1000
B(Purchase) = 100
V(Purchase, price) = 100, ranging from 0 to 200, with all values equally likely

T(Customer) = 3000
B(Customer) = 200
V(Customer, custId) = ~~50~~ 3000 (clarification during exam)

Number of memory pages available = 20

a) (6 points) Given this query:

```
SELECT *
FROM   Purchase p, Customer c
WHERE  p.custId = c.custId AND p.price < 100 AND c.custId = 42
```

Indicate if each of the indexes below can help speeding up query execution, assuming that it is the only index available. For each below you get 1 point for answering it correctly, -0.5 point for an incorrect answer, and 0 point for no answer. The minimum you will get for this problem is 0.

1)	Hashtable index on Purchase(price)	Yes	**No**
2)	B-tree index on Purchase(pid, price)	Yes	**No**
3)	Hashtable index on Customer(custId)	**Yes**	No
4)	Hashtable index on Purchase(custId)	**Yes**	No
5)	B-tree index on Purchase(price, pid)	**Yes**	No
6)	Hashtable index on Purchase(price, pid)	Yes	**No**

b) Which join algorithm would you use to execute the join in a) to minimize execution time? Assume that there are no indexes available. Be clear about how the join will be executed, i.e., what attribute will you sort on if sorting is involved, what relation will you construct a hashtable on if one is needed, etc. Briefly explain why. (8 points)

We will first perform selection to reduce the size of the input tables. Evaluating the selection on p.price will result in 100 * (100/200) = 50 pages (call this T1), and evaluating the selection on c.custId will result in 200 * 1/3000 = 1 page (call this T2). Since we don't have enough pages in memory to hold both T1 and T2, we don't want to perform the join using sort-merge. We can either do nested loop join (with T2 being the outer loop relation that always reside in memory), or hash join after constructing hashtable on T2.

c) Are these two queries semantically equivalent?

Query 1:
```
SELECT COUNT(*) FROM A
 WHERE A.a IN (SELECT x FROM B WHERE B.c = A.b)
```
Query 2:
```
SELECT COUNT(*) FROM A, B
 WHERE A.a = B.x AND B.c = A.b
```

If equivalent, write "equivalent" below. If not, write "not equivalent," and describe the contents of A and B such that the two queries output different results. (8 points)

Not Equivalent. Consider A(a,b) = (1,1) and B(x,c) = (1,1), (1,1).
Here the second query produces 2 results while the first query produces 1 result. Note that query 1 is a correlated subquery, which means that the inner query is evaluated once per record in A, supplying the A.b value and then checking to see if the corresponding A.a value satisfies the IN clause.

d) Describe two differences between how relational and semi-structured data models manage data instances. (8 points)

Many possibilities. For instance:

- Encode instances using relations rather than structured key-value pairs
- Arbitrary nesting of key-value pairs rather than flattening into multiple rows
- Storing collections directly as attributes rather than requiring flattening

e) Would using an unclustered index ever perform worse than a sequential scan on the same table? If yes, describe a scenario for which it is true, otherwise write "no" below. (8 points)

When using the select operator to scan a table R, sequential scans have a cost of B(R), where using unclustered index has cost T(R) * X, where X is the selectivity. If T(R) * X > B(R), or in general if the number of distinct tuples is small, then using an unclustered index will be more costly than sequential scan.

-- END OF EXAM --