CSE344 Midterm Exam
Winter 2017
February 13, 2017

- Please read all instructions (including these) carefully.
- **This is a closed-book exam. You are allowed a one-page cheat sheet that you can write on both sides.**
- 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 5 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 blank pages as scratch paper. **Do not** use any additional scratch paper.
- Relax. You are here to learn. 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_____ SECTION: _____

STUDENT NUMBER: _____

Problem	Points
1	20
2	42
3	38
Total	100

Problem 1: Warm up (20 points total)

Select either True or False for each of the following questions. For each question you get 2 points for answering it correctly, -1 point for an incorrect answer, and 0 point for no answer. The minimum you will get for this entire problem is 0.

a) Each relation can have multiple clustered or unclustered indexes.
[explanation: can't have multiple clustered indexes]

True False

b) A Datalog query without negation can only express monotone queries.

True False

c) Sequential scan of the entire relation R costs at least T(R).
[explanation: cost at least B(R)]

True False

d) A Datalog query is safe if all variables appear in relational atoms.
[explanation: **positive** relational atoms]

True False

e) Relational calculus queries that use universal quantifiers are always monotonic.
[explanation: mostly non-monotonic]

True False

f) Equijoin checks the equality of all common attributes between two relations.
[explanation: that's natural join]

True False

g) All aggregations can only be applied to a single attribute.
[explanation: e.g., sum(attribute1+attribute2)]

True False

h) Secondary indexes should always be dense in order to be utilized for query optimization.

True False

i) Hash join always runs more efficiently than sort-merge join.
[explanation: they can have the same runtime, e.g., if both relations are already in memory]

True False

j) Key-value stores use the relational data model to store data.
[explanation: they use non-relational models]

True False

Problem 2: Writing Queries (42 points total)

Write the following queries using the schema below. Datalog and relational calculus queries are evaluated using set semantics, but SQL and relational algebra will be evaluated using bag semantics. While we are not asking for the most efficient solution, but we reserve the right to take off points if your solution is overly redundant or unnecessarily inefficient.

```
Class (dept, number, quarter, title)
Student (username, first, last, year)
Takes (username, dept, number, quarter, grade)
-- For the Takes relation:
-- grade = -1 for current quarter classes
-- username is foreign key to Student
-- (dept, number, quarter) is foreign key to Class
```

a) Write a SQL query that returns the *quarter* when students with username "alex" and "bob" take at least 1 class together , along with the *number of classes* that they have taken together in that quarter. (11 points)

```
SELECT t1.quarter, COUNT(*)
FROM Takes t1, Takes t2
WHERE t1.username = "alex" AND t2.username = "bob"
AND t1.dept = t2.dept AND t1.number = t2.number
AND t1.quarter = t2.quarter
GROUP BY t1.quarter
```

b) Write a domain independent relational calculus query that returns the usernames of the students who take *exactly two* classes in the "CSE" department in the "17wi" quarter. (11 points)

```
P(u) = ∃f,l,y .  Student(u, f, l, y)  ∧
       ∃n1,g1,n2,g2 . Takes(u, 'CSE', n1, '17wi', g1) ∧
                      Takes(u, 'CSE', n2, '17wi', g2) ∧ n1 != n2 ∧
       ∀n3,g3 . Takes(u, 'CSE', n3, '17wi', g3) ⇒ (n1 = n3 ∨ n2 = n3)
```

Schema duplicated here for your convenience.

```
Class (dept, number, quarter, title)
Student (username, first, last, year)
Takes (username, dept, number, quarter, grade)
-- For the Takes relation:
-- grade = -1 for current quarter class
-- username is foreign key to Student
-- (dept, number, quarter) is foreign key to class
```

c) Write a safe Datalog + negation program that returns the title of all classes which have not been taken by any first year student. (10 points)

```
Taken(d, n, q):- Takes(x, d, n, q, _), Student(x,_,_,1)
Answer(t):- Class(d, n, q, t), not Taken(d, n, q)
```

d) Convert the following relational algebra query to SQL and explain what it computes in one sentence. (10 points)

$$\Pi_{first,last}((\Pi_{username}(Student) - \Pi_{username}(Student \bowtie \sigma_{quarter = \text{“17wi”}}(Takes))) \bowtie_{username=username} Student)$$

```
SELECT S.first, S.last
FROM Student S
WHERE S.uid NOT IN (SELECT T.uid
                    FROM Takes T
                    WHERE T.quarter = ‘17wi’)


Returns the first and last name of all the students who have not taken any
classes in ‘17wi’ quarter.
```

Problem 3: Short Questions (38 points total)

a) (8 points) Given this query using the schema from problem 2:

```
SELECT S.last FROM Student S WHERE 3 > year AND year > 1
```

Circle the top **ONE** index that would be most useful in reducing query processing time. (There can be more than one answer, but only circle one. You will get 0 points if the wrong or multiple entries are circled). Assume that there are no existing indexes and data is stored using heap files. Ignore the time needed to construct the indexes.

Hashtable on `Student(year)`	B+ tree on `Student(year)`
Hashtable on `Student(last)`	B+ tree on `Student(last)`
Hashtable on `Student(year, last)`	B+ tree on `Student(year, last)`
Hashtable on `Student(last, year)`	B+ tree on `Student(last, age)`

[Answer: *B-tree index on Student(year, last)* for full points, *B-tree index on Student(year)* for half points, 0 points for everything else.**]**

b) (8 points) Describe one benefit of the semi-structured data model over the relational model.

Many possibilities: easier to scale up, simpler data model and query language, etc.

c) (8 points) Indicate whether each relational algebra expression below is equivalent to the following SQL query. You get 2 points for each correct answer, -1 point for each incorrect one, and 0 for no answer. Minimum you will get for this question is 0.

```
SELECT R.a, S.c FROM R, S WHERE R.b = S.b AND R.b > 10
```

i) $\Pi_{A,C}(\sigma_{R.b>10}(R) \bowtie_{R.b=S.b} \sigma_{S.b>10}(S))$ — Yes / No: **Yes**

ii) $\Pi_{A}(\sigma_{R.b>10}(R)) \times \Pi_{C}(\sigma_{S.b>10}(S))$ — Yes / No: **No**

iii) $\Pi_{A,C}(\sigma_{S.b<>null\ and\ R.b>10}(R \bowtie_{R.b=S.b} S))$ — Yes / No: **No**

iv) $\Pi_{A,C}(\sigma_{R.b=S.b}(R \times (\sigma_{S.b>10}(S))))$ — Yes / No: **Yes**

d) (6 points total) Given the following statistics, indicate whether having an unclustered index would ever *perform worse* than doing a sequential scan for the query. You will get 2 points for each correct answer, -1 point for incorrect one, and 0 for no answer. Minimum you will get for this question is 0.

```
SELECT R.a FROM R WHERE R.a = 42
```

i) B(R) = 6,000, T(R) = 100,000, V(R, a) = 30	Yes	<u>No</u>
ii) B(R) = 3,000, T(R) = 600,000, V(R, a) = 300	Yes	<u>No</u>
iii) B(R) = 3,000, T(R) = 100,000, V(R, a) = 30	<u>Yes</u>	No

e) (8 points) Express the relational algebra δ operator on a relation R using other (base or extended) relational algebra operators, or write "N/A" if it is not possible to do so.

$\delta(R) = \gamma_{A1,A2,...,An}(R)$
where A1,A2,...,An are the attributes of R

-- END OF EXAM --