CSE 344 Midterm

November 9, 2011, 9:30am - 10:20am

Name: _____

Question	Points	Score
1	40	
2	40	
3	20	
Total:	100	

- This exam is open book and open notes but NO laptops or other portable devices.
- You have 50 minutes; budget time carefully.
- Please read all questions carefully before answering them.
- Some questions are easier, others harder. Plan to answer all questions, do not get stuck on one question. If you have no idea how to answer a question, write your thoughts about the question for partial credit.
- Good luck!

1. **SQL and Physical Tuning** (40 points)

   You are in charge of managing the program committee for an important conference. The following database stores information about papers submitted to the conference (table `Paper`), reviewers on the program committee (table `Reviewer`), and the assignment of reviewers to papers (table `Reviews`). Each reviewer on the program committee will have to review a set of papers. Each paper will be reviewed by some subset of reviewers.

   ```
   Paper(pid, title)
   Reviewer(rid, name)
   Reviews(rid, pid)
   ```

   - `pid` is a unique paper identifier and the primary key of the `Paper` table.
   - `rid` is a unique reviewer identifier and the primary key of the `Reviewer` table.
   - `Reviews.rid` is a foreign key that references `Reviewer.rid`.
   - `Reviews.pid` is a foreign key that references `Paper.pid`.
   - A reviewer is assigned zero or more papers.
   - A paper is assigned zero or more reviewers.

   (a) (15 points) Write a **SQL query** that finds all papers with fewer than three reviewers assigned to them. The output of the query should be a list of paper titles. The result should **include papers without any reviewers** assigned to them.

   Answer (write a SQL query):

   Solution:

   ```
   SELECT P.title
   FROM Paper P LEFT OUTER JOIN Reviews X ON P.pid = X.pid
   GROUP BY P.pid, P.title
   HAVING count(*) < 3
   ```

```
Paper(pid, title)
Reviewer(rid, name)
Reviews(rid, pid)
```

(b) (15 points) Write a **SQL query** that finds the reviewers with the most papers assigned to them. There can be more than one such reviewer. The output of the query should be a list of reviewer names. A reviewer should be listed if no other reviewer has strictly more papers to review.

Answer (write a SQL query):

Solution:

```
SELECT R.name
FROM Reviewer R, Reviews X1
WHERE R.rid = X1.rid
GROUP BY R.rid, R.name
HAVING count(*) >= ALL ( SELECT count(*)
                         FROM Reviews X2
                         GROUP BY X2.rid )
```

```
Paper(pid, title)
Reviewer(rid, name)
Reviews(rid, pid)
```

(c) (10 points) Suggest **2 indexes** that would speed-up your queries from the previous questions. **Explain** why you are selecting these indexes.

Answer (Suggest and justify the selection of two indexes):

Solution: Since the above queries join the three tables together in different ways, a natural choice is to index the primary keys and foreign keys. One possible solution is to index `Paper.pid` and `Reviewer.rid`. Another option is to index `Reviews.pid` and `Reviews.rid`. The indexes do not need to be clustered since the queries are not doing any range selections on these attributes. [Other solutions are possible].

2. **Relational Algebra, Calculus, and Datalog** (40 points)

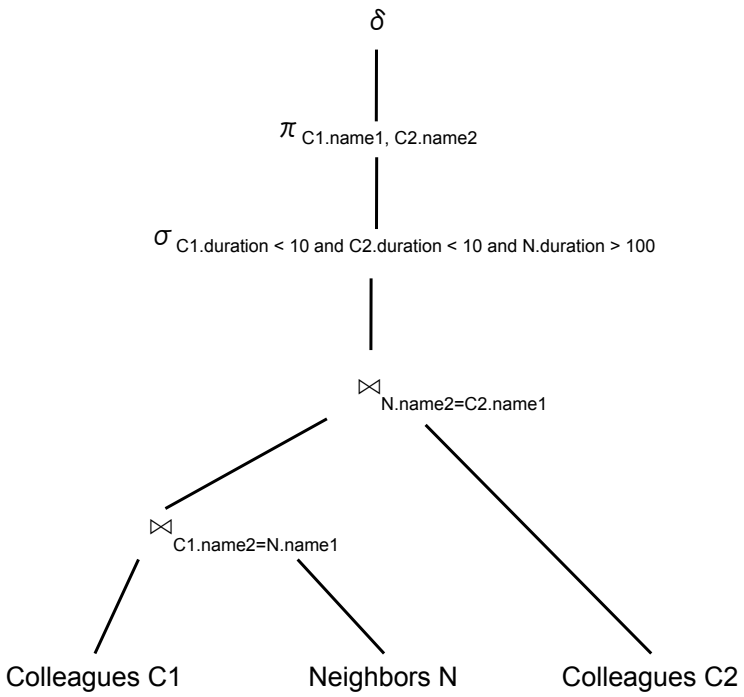
Consider the following database schema:

```
Neighbors(name1,name2,duration)
Colleagues(name1,name2,duration)
```

(a) (15 points) Write a **Relational Algebra Plan** for the SQL query below. Your answer can be in the form of an expression or a tree, whichever you prefer:

```
SELECT DISTINCT C1.name1, C2.name2
FROM Colleagues C1, Neighbors N, Colleagues C2
WHERE C1.name2 = N.name1
AND   N.name2 = C2.name1
AND   C1.duration < 10
AND   C2.duration < 10
AND   N.duration > 100
```

Answer (write a Relational Algebra Plan):



Solution:

```
Neighbors(name1,name2,duration)
Colleagues(name1,name2,duration)
```

(b) (15 points) Write a **Datalog query** that returns all neighbors who do not have any colleagues in common.

Answer (write a Datalog query):

Solution:
NonAnswers(n1,n2) :- Neighbors(n1,n2,_), Colleagues(n1,c,_), Colleagues(n2,c,_)
A(n1,n2) :- Neighbors(n1,n2,_) NOT NonAnswers(n1,n2)

```
Neighbors(name1,name2,duration)
Colleagues(name1,name2,duration)
```

(c) (10 points) Indicate if the following **relational calculus** queries are correct or not (true or false). Note: This is not meant to be a tricky question. Errors, if any, should be reasonably obvious. You do NOT need to correct wrong queries:

Answer (true or false)

- Find all people who have a neighbor that has a colleague.

  $A(x) = \exists y.\exists z.Neighbors(x, y, -) \wedge Colleagues(y, z, -)$

  TRUE/FALSE

- Find all people who have only neighbors that are also their colleagues.

  $A(x) = Neighbors(x, -, -) \wedge (\forall y.Neighbors(x, y, -) \wedge Colleagues(x, y, -))$

  TRUE/FALSE

- Find all people who have only neighbors that have at least one colleague.

  $A(x) = Neighbors(x, -, -) \wedge (\forall y.Neighbors(x, y, -) \Rightarrow \exists z.Colleagues(y, z, -))$

  TRUE/FALSE

Solution:
TRUE
FALSE. The correct answer is
$A(x) = Neighbors(x, -, -) \wedge (\forall y.Neighbors(x, y, -) \Rightarrow Colleagues(x, y, -))$
TRUE

3. **XML, XPath, and XQuery** (20 points)

Consider the following DTD, which describes the schema for a database containing a list of players. The rank of a player indicates whether the player is a "Beginner" or an "Advanced" player. The score is the total number of points accumulated by the player.

```
<!DOCTYPE game [
    <!ELEMENT game (player*)>
    <!ELEMENT player (rank,score)>
    <!ATTLIST player name CDATA #REQUIRED >
    <!ELEMENT rank (#PCDATA )>
    <!ELEMENT score (#PCDATA )>
]>
```

(a) (20 points) Write an XQuery expression that reformats a valid XML document, as per the above DTD, into one that matches the following DTD. In this new format, we want to group "Beginner" players into one category and "Advanced" players into another category:

```
<!DOCTYPE game [
    <!ELEMENT game  (category*)>
    <!ELEMENT category (rank, player*)>
    <!ELEMENT player (name, score) >
    <!ELEMENT rank     (#PCDATA )>
    <!ELEMENT name     (#PCDATA )>
    <!ELEMENT score    (#PCDATA )>
]>
```

Answer (write an XQuery):

Solution:

```
<game>
 { for $d in doc("game.xml")/game
   for $r in distinct-values($d/player/rank/text())
   return
    <category>
        <rank> {$r} </rank>
        { for $p in $d/player[rank=$r]
          return
            <player>
              <name> { string($p/@name) } </name>
              <score>{ $p/score/text() } </score>
            </player>
        }
    </category>
 }
</game>
```