CSE 444 Final --- Winter 2005

March 16, 2005 2:30 - 4:20 p.m.

Name_____

Total number of points: 100

Total time: 1h 50'

This is an open book exam.

1. [35 points] Consider the following relational schema:

Person(<u>name</u>, age, city)
Trusts(name1, name2)

The Trusts relation tells us who trusts whom. This relation is not necessarily symmetric: i.e., if 'John' trusts 'Mary', then it is not necessarily the case that 'Mary' trusts 'John'. This relation is not necessarily transitive: i.e., if 'John' trusts 'Mary' and if 'Mary' trusts 'Fred', it is not necessarily the case that 'John' trusts 'Fred'.

a. [15 points] For each pairs of queries Q1, Q2 below, indicate whether Q1 is 'contained' in Q2, i.e. if every answer returned by Q1 is also returned by Q2. You have to answer **contained** or **not contained** to each question. For example, if Q1 and Q2 were:

```
Q1:  select distinct x.city
     from Person x, Trusts y
     where x.age > 40 and x.name = y.name1
Q2:  select distinct x.city
     from Person x
     where x.age > 20
```

then you would say that Q1 is **contained** in Q2. However, if Q1 and Q2 were:

```
Q1:  select distinct x.city
     from Person x, Trusts y
     where x.name = y.name1 and y.name2 = 'Fred'

Q2:  select distinct x.city
     from Person x
```

then you would say that Q1 is **not contained** in Q2.

i.

Q1:
```
select distinct x.name1
from Trusts x, Trusts y
where x.name2 = y.name1 and
      y.name2='Betty'
```

Q2:
```
select distinct x.name1
from Trusts x
where x.name2='Betty'
```

ii.

Q1:
```
select distinct x.name1
from Trusts x
where x.name2 = 'Fred'
```

Q2:
```
select distinct x.name1
from Trusts x, Trusts y, Trusts z
where x.name2 = y.name2 and
    y.name1 = z.name1 and
    z.name2 = 'Fred'
```

iii.

Q1:
```
select x.name, count(*)
from Person x, Trust y
where x.city = 'Seattle' and
   x.age > 25 and
   x.name = y.name1
group by x.name
```

Q2:
```
select x.name, count(*)
from Person x, Trust y
where x.city = 'Seattle' and
      x.name = y.name1
group by x.name
```

iv.

Q1:
```
select x.city, count(*)
from Person x, Trust y
where x.city = 'Seattle' and
   x.age > 25 and
   x.name = y.name1
group by x.city
```

Q2:
```
select x.city, count(*)
from Person x, Trust y
where x.city = 'Seattle' and
      x.name = y.name1
group by x.city
```

b. [20 points]

A "loner" is a person who trusts no one but himself.
A "loyal" is a person who trusts only those who trust him.
A "ruler" is a person who trusts only those who trust only him.

Indicate for each of the queries below whether it computes the loners, or the loyals, or the rulers, or none of the above:

i.

```
select distinct x.name1
from Trusts x
where x.name1 not in
          (select y.name1
           from Trusts y
           where y.name2 in
                    (select z.name1
                     from Trusts z
                     where z.name2 != y.name1))
```

ii.

```
select distinct x.name1
from Trusts x
where x.name2 not in
            (select y.name2
             from Trusts y
             where x.name1=y.name1
                    and y.name2 != x.name1)
```

iii.

```
select distinct x.name1
from Trusts x
where x.name1 in
             (select y.name1
              from Trusts y
              where y.name2 not in
                        (select z.name1
                         from Trusts z
                         where z.name2 = y.name1))
```

iv.

```
select distinct x.name1
from Trusts x
where x.name1 not in
            (select y.name1
             from Trusts y
             where y.name2 not in
                        (select z.name1
                         from Trusts z
                         where z.name2 = y.name1))
```

2. [20 points] Consider an XML document containing information about job postings and job applications. The postings are grouped by company, and applications are listed under postings. An application contains only the applicant's name. For example:

```
<jobs>
   <company>
      <name> MicroScience Corp. </name>
      <posting>
         <jobtitle> sales rep </jobtitle>
         <salary> 30000 </salary>
         <application> Mark </application>
         <application> Lucy </application>
         <application> Frank </application>
      </posting>
      <posting>
         <jobtitle> technology consultant </jobtitle>
         <salary> 80000 </salary>
         <application> Lucy </application>
         <application> Fred </application>
         <application> Mark </application>
         <application> Betty </application>
      </posting>
   </company>
   <company>
      <name> MacroConsulting Inc. </name>
      <posting>
         <jobtitle> technology consultant </jobtitle>
         <salary> 40000 </salary>
         <application> Frank </application>
      </posting>
      <posting>
         <jobtitle> debugger </jobtitle>
         <salary> 20000 </salary>
      </posting>
      <posting>
         <jobtitle> programmer analyst </jobtitle>
         <salary> 35000 </salary>
         <application> Lucy </application>
         <application> Mark </application>
      </posting>
   </company>
</jobs>
```

a. [10 points] For each of the XPath expressions, indicate how many answers it will return on the example XML document on previous page. For example, if the XPath expression is

/jobs/company[name/text()='MacroConsulting Inc.']/posting

then you will answer 3. For each question, you have to turn in a number.

i. //application

ii. /jobs/company/posting[salary/text()>60000]//application

iii. /jobs/company[posting/salary/text()>60000]//application

b. [10 points] Write an XQuery expression that returns the names of all applicants who submitted applications to at least two companies. Your query should a list of **<name>** elements, and each element should be included only once. For example, if your query were to run on the XML document shown above, then it will return:

<name> Mark </name>
<name> Lucy </name>
<name> Frank </name>

3. [35 points]

a. [15 points] Consider a transaction T consisting of the following actions:

START, R(A), R(B), W(A), W(B), COMMIT

This generates in the following log entries:

<START T>
<T, A, 10>
<T, B, 20>
<COMMIT T>

We examine several sequences of *write* operations to disk. Each line represents one disk write, e.g. <T, A, 10> represents the operation that writes this log entry to disk, while OUTPUT(A) represents writing the A element to disk.

S1: <START T>
OUTPUT(A)
OUTPUT(B)
<T, A, 10>
<T, B, 20>
<COMMIT T>

S2: <START T>
<T, A, 10>
<T, B, 20>
OUTPUT(B)
OUTPUT(A)
<COMMIT T>

S3: <START T>
<T, A, 10>
<T, B, 20>
<COMMIT T>
OUTPUT(A)
OUTPUT(B)

S4: <START T>
<T, A, 10>
OUTPUT(A)
<T, B, 20>
OUTPUT(B)
<COMMIT T>

i. [5 points] Assume that the transaction manager uses an undo log. Which of the above sequences is legal, according to the rules of the undo log? You have to turn in a list of sequence names, e.g. **S1, S3, S4.**

ii. [5 points] Assume that the transaction manager uses a redo log. Which of the above sequences is legal, according to the rules of the redo log?

iii. [5 points] Assume that the transaction manager uses a undo/redo log. (Assume that the entries <T, A, 10> and <T, B, 20> are replaced with <T, A, 1, 10> and <T, B, 2, 20>). Which of the above sequences is legal, according to the rules of the undo/redo log?

b. [20 points] Consider the following schedule:

START(T1), START(T2), $R_1(A)$, $R_2(B)$, $W_1(B)$, $W_2(C)$, COMMIT(T1), COMMIT(T2)

i. [4 points] Draw the precedence graph, and indicate whether the schedule is serializable. For this question you will ignore the START and COMMIT actions, and consider only the R and W actions.

ii. [8 points] Assume a lock-based concurrency control manager, which automatically introduces a lock action right before each read or write, and introduces all unlock actions before commit.

— Indicate up to which point the schedule can execute, by drawing a line below after the last action that can execute:

START(T1), START(T2), $R_1(A)$, $R_2(B)$, $W_1(B)$, $W_2(C)$, COMMIT(T1), COMMIT(T2)

— Indicate the subsequent actions, in the order in which they are scheduled by the lock-based concurrency control manager. You have to turn in a sequence of actions, e.g. COMMIT(T2), $W_2(C)$.

iii. [8 points] Assume a concurrency control manager based on timestamps.

— Indicate up to which point the schedule can execute, by drawing a line below after the last action that can execute:

START(T1), START(T2), $R_1(A)$, $R_2(B)$, $W_1(B)$, $W_2(C)$, COMMIT(T1), COMMIT(T2)

— Indicate the subsequent actions, in the order in which they are scheduled by the concurrency control manager based on timestamps.

4. [10 points] Query execution. Consider two tables R(A, B) and S(C, D), and the following statistics:

B(R) = 5
T(R) = 20
B(S) = 100
T(S) = 400
V(S, C) = 50
M = 1000

There is a clustered index on S.C. Consider the logical operator $R \rhd\lhd_{B=C} S$, and the following two physical operators that can be used to implement it:

P1 = main memory hash-join
P2 = index join

Compute for each of the number of disk I/O's required to execute the operator. You have to turn in two numbers, e.g. COST(P1)=42245, COST(P2)= 7.