Database Systems, CSCI 4380-01 Exam #3
Monday May 9, 2010 at 2 pm

1 (20)	2 (20)	3 (20)	4 (20)	5 (20)	6 (20)	TOTAL (100)

Note. The exam is open book and open notes. Use your own book and notes only, sharing is not allowed. Electronic gadgets are NOT allowed during the exam. Write your answers clearly, legibly and explain your reasoning as much as you can. If I cannot read or understand your answers, you will not get points.

There are 6 questions in this exam. Answer any 5 of them. If you answer all 6, I will only read the first 5 answers. Mark clearly which question you do not want me to grade.

In SQL queries in Question 5, do not use views (i.e. using CREATE VIEW statements), triggers and other procedural elements. Make sure you use DISTINCT only when you have to.

Question 1 (20 points). You are given the following schedule:

$r_1(X)\ r_1(Z)\ r_2(Z)\ w_2(Z)\ r_3(W)\ w_3(X)\ w_1(Y)\ r_4(K)\ r_4(W)\ w_4(Z)\ w_4(Y)$
$commit_1\ commit_2\ commit_3\ commit_4$

(a) List all conflicts in this schedule and draw the conflict diagram. Is this schedule is serializable? Why or why not? If this schedule is serializable, then find a serial schedule that is equal to this schedule.

Answer. Conflicts: $r_1(X), w_3(X),\ r_1(Z), w_2(Z)\ ,\ r_1(Z), w_4(Z),\ r_2(Z), w_4(Z),\ w_2(Z), w_4(Z),$ $w_1(Y), w_4(Y)$.

Graph $T_1 \rightarrow T_2, T_3, T_4,\ T_2 \rightarrow T_4$. No cycles. An equivalent serial schedule is : T_1, T_2, T_3, T_4 (there are actually 3 possible serial schedules).

(b) Is this schedule possible under the two phase locking scheme if all items are locked with a single type of lock? Explain why or why not in detail.

Answer. Yes, as long as T_1 locks Y before it unlocks Z, this schedule is possible. The following is one such possible sequence of locks.

$l_1(X)r_1(X)l_1(Z)l_1(Y)r_1(Z)u_1(Z)l_2(Z)r_2(Z)$
$w_2(Z)l_3(W)r_3(W)u_1(X)l)3(X)w_3(X)w_1(Y)u_1(Y)l_4(K)r_4(K)$
$u_3(W)l_4(W)r_4(W)u_2(Z)l_4(Z)w_4(Z)l_4(Y)w_4(Y)$
$commit_1 commit_2 commit_3 commit_4$

(c) True or false: If a schedule is serializable and we are using a single type of locks, no other transaction can read the data written by another transaction X before X commits. Explain why or why not.

Answer. False. As long as the transaction releases the lock before committing, the other transaction can read the value written by that the transaction. Example $r_1(X)w_1(X)r_2(X)commit_1 commit_2$ is serializable.

Question 2 (20 points each). Answer the following questions:

(a) In order to avoid reading the log completely from the beginning during recovery, how can we modify the log? Explain briefly.

Answer. We can add a checkpoint to the log which would list the active transactions and dirty pages. Then, the recovery needs to first locate the last checkpoint and start from the first uncommitted change according to the dirty page table stored in the checkpoint.

(b) Suppose you are using the following protocol for transaction management:

When a transaction wants to commit, a commit record is written. But, the transaction is not yet allowed to commit.

The transaction manager periodically flushes the log to disk completely, then writes all the changed data pages to disk. When both of the write operations complete, all transactions that had written a commit log record are allowed to commit.

Under this protocol, if a crash occurs, is there a need to do REDO and/or UNDO? Explain why.

Answer. Both are needed. REDO is needed because a crash can occur after writing the log to disk but before storing all the changed pages. UNDO is required because data pages can be written to disk before the transaction commits, as long as the log is written first.

(c) If you are using REDO/UNDO recovery, and the following are the contents of the log and the disk after crash, which log entries should be redone and which log entries should be undone? Explain why briefly.

LOG:

LSN	Entry
10	T1 update P1
11	T2 update P2
12	T2 update P3
13	T3 update P4
14	T1 update P5
15	T1 commit

DATA:

Data Page	LSN of Last recorded log entry
P1	10
P2	5
P3	12
P4	13
P5	6

Answer. We must try REDO of 10,14 because T1 committed. Only 14 is necessary. We must try UNDO of 13,12,11 as T2 and T3 never committed. But, 11 was not committed, so it can be skipped.

The final operations are: REDO 14, UNDO 13, UNDO 12, in this order.

Question 3 (20 points). You are given the following statistics for `Students(id, gender, class)` with 500 tuples and `Transcript(studid, crn, semester, year, grade)` with 10,000 tuples:

Attribute	Values	Min	Max
Students.id	500	66001234	66119999
Students.gender	2	'F'	'M'
Students.class	5	1	5
Transcript.studid	480	66001234	66119999
Transcript.crn	50	12345	23456
Transcript.semester	3	1	3
Transcript.year	8	2004	2011
Transcript.grade	10	'A'	'W'

```
Q1.
SELECT
  *
FROM
  transcript
WHERE
  semester = 1
  AND year = 2010 ;

Q2.
SELECT
  *
FROM
  students
WHERE
  gender = 'F'
  AND class = 2;

Q3.
SELECT
  *
FROM
  students S
  , transcript T
WHERE
  S.id = T.studId
  and T.grade = 'A'
```

(a) Find the total number of expected tuples in the above queries Q1, Q2 and Q3 (write your answer next to each query above).

Answer. Q1: $10,000 * 1/3 * 1/8 = 417$.

Q2: $500 * 1/2 * 1/5 = 50$.

Q3: $10,000 * 500 * 1/500 * 1/10 = 1,000$

(b) What is the cost of answering query Q2 above using an index on $Student(class)$ with height 1 (root and leaf level), and 40 leaf nodes?

Answer. $1 + 40/5 + 500/5$, for 1 root, 40/5 leaf nodes and 100 tuples from the relation to find the tuples with $class = 2$ and check for gender and return the attributes. The 100 tuples can in the worst case be in 100 different pages.

Question 4 (20 points). Answer the following given $PAGES(R) = 300$, $PAGES(S) = 1,000$ and index on $R(A)$ with height 1 (root and leaf) and 30 nodes at the leaf level.

(a) What is the cost of sorting relation R with $M = 10$ pages?

Answer. 600 (create 30 sorted groups) + 600 (merge 30 into 3 groups) + 300 (merge 3 groups and finish sort) = 1,500

(b) What is the cost of joining $R \bowtie S$ using block-nested-loop join with $M = 51$ pages (R is the outer relation and S is the inner relation)?

Answer. $300 + 300/50 * 1,000 = 6,300$

(c) What is the cost of the query plan given in the figure below (index look up on $R.A$ followed by a block nested loop join on attribute A)? Note: Yes, you really have all the statistics you need for this question.

Answer. First, given we only need attribute A for this query, we can simply scan the index and find the matching tuples. We know that information we need is at most 30 pages (the whole leaf level). However, given the condition on A, we actually expect about 1/3 of the index to match. So, at most, the matching tuples will fit in 10 pages.

The join with S will allocate 50 pages to R, which means all the tuples from R (10 pages) will fit memory. So, we only have to read S 1 times, cost of 1,000 pages.

What is the cost of reading R? We need to read the root node (which you can disregard assuming it is already in memory) and the cost of scanning a 1/3 of the index. Hence, of reading $R = 1 + 30 * 1/3 = 11$. Cost of join, requires reading S $ceil(10/50) = 1$ times. Total cost $= 11 + 1,000 = 1,011$

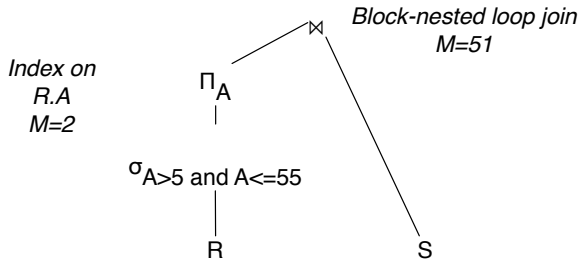

Figure 1: The query plan for question 4

Data Model.

Country(id, name)
Person(id, name, gender, birthday, birthCountryId, fatherId, motherId)
CitizenOf(personId, countryId)

In Person, fatherId and motherId are foreign keys to Person(id), but they can also be null if the values are not known. The attribute birthCountryId is a foreign key to Country.

In CitizenOf, personId is foreign key to person and countryId is a foreign key to Country. This relation stores which countries a person is a citizen of (some people may be dual or triple citizens).

Question 5 (20 points). Suppose you are given the data model above. Write the following queries using SQL:

(a) Find people who have a mother and father with citizenship from completely different countries (if mother and father are known). Return their id of the found people.

Answer.

```
select
    id
from
    person p
where
    not exists
      (select
            1
       from
            citizenOf c1
            , citizenOf c2
       where
            c1.personid = p.motherid
            and c2.personid = p.fatherid
            and c1.countryid = c2.countryid) ;
```

(b) Find people who have dual citizenship (exactly 2 countries, no more), and are born in a country different than either of their current countries of citizenship. Return their id.

Answer.

```
select
    id
from
    person p
    , citizenOf c
where
    p.id = c.personid
    and p.birthCountryid not in
         (select
            c2.countryId
          from
            citizenOf c2
          where
            c2.personid = p.id)
group by
```

```
     p.id
having
     count(*) = 2 ;
```

Question 6 (20 points). Using the data model from Question 5, write a pl/pgsql procedure (or a procedure in pseudo-code that combines SQL queries with procedural code) for the following: Given the id of a person, find which generation of US citizen he/she is.

If she is not a US citizen, then she belongs to generation 0.

If she is a US citizen, but neither of her parents are US citizens, then she is of generation 1.

Similarly, any person who is a US citizen with parents belonging to generation i belongs to generation i+1.

If the parents of a person are not known, assume they are not US citizens.

For example, if the person is a US citizen and her parents are US citizens, but none of her grand parents are US citizens, then she is generation 2.

Hint. Finding which generation everyone is in first may make things much easier.

Answer.

```
procedure x(qid int) ;
declare
    usaid INT ;
    oldcnt INT ;
    newcnt INT ;
    qgen INT ;
begin
    select id into usaid from country where name = 'USA' ;
    create table gen(id int, cnt int) ;
    insert into table gen
        select id, 0
        from person
        where id not in
           (select c.personid from citizenOf c where c.countryid = usaid) ;

    curGen = 1;
    select count(*) into oldcnt from gen ;

    insert into table gen
      select p.id, curGen
      from person p
      where id in (select c.personid from citizenOf c where c.countryid = usaid)
            and motherid not in (select id from gen)
            and fatherid not in (select id from gen) ;

    loop
        insert into table gen
          select p.id, 1 +
                       (select max(cnt)
                        from gen
                        where g.id = p.motherid or g.id = p.fatherid)
          from person p
          where id in (select c.personid from citizenOf c
                       where c.countryid = usaid)
                and id not in (select id from gen)
                and (exists (select 1 from gen g
                             where g.id = p.motherid) or p.motherid is null)
```

```
                and (exists (select 1 from gen g
                            where g.id = p.fatherid) or p.fatherid is null)

        select count(*) into newcnt from gen ;

        if newcnt = oldcnt then
           break loop ;
     end loop ;

     select cnt into qgen from gen where id = qid ;
     drop table gen ;
     return qgen ;

end ;
```