Prof. Li-Yan Yuan

CMPUT 391: Database Management Systems

Midterm Examination

October 23, 2002

It is a close-book examination and the time for the test is 50 minutes. There are seven (7) questions with 15 points for each of the first 6 questions, and 10 points for the last one. Good luck to all of you.

1. Consider the table created by the following statements:

```
CREATE TYPE course_type AS OBJECT (
   course_id  VARCHAR(20),
   name       VARCHAR(20)
);

CREATE TABLE course OF course_type;
```

   Assume that *course* has been populated with several rows, including one whose course_id is '391'.

   Use SQL-99 or Oracle statements to

   (a) create a table, called *registration*, that contains three columns: student_id (a VARCHAR(20) type), course (a reference type to course_type), and grade (an integer type).

   (b) insert a new row to *registration* which indicates that a student whose id is '1234' is taking '391' with grade 0.

   Solution:

```
CREATE TABLE registration (
   student_id varchar(20),
   course  REF course_type,
   grade int
)

insert into registration
 select '1234', REF(cc), 0
 from course cc
 where cc.course_id = '391';

Or,

insert into registration values
('1234',
 (select REF(cc)from course cc
 where cc.course_id = '391'),
 0
)
```

2. Consider a database consisting of the following tables.

```
employee( e_id, e_name, position, salary )
works( e_id, d_id )
dept( d_id, budget, manager_id )
```

The first table indicates the job title and salary for each employee, the second shows the department an employee works for, and the last one indicates the budget and boss of each department.

Write one or more SQL-99 triggers to ensure that whenever an employee is given a raise, the manager's salary must be increased with the same amount. (No wonder why so many employees wish to be a manager).

This can be done by the following trigger.

```
CREATE TRIGGER manager_raise
AFTER UPDATE OF salary ON employee
FOR EACH ROW
BEGIN
  if ( new.salary -old.salary ) > 0
    UPDATE employee
    SET salary = salary
               + (new.salary-old.salary)
    WHERE  e_id in ( SELECT d.manager_id
                     FROM dept d, works w
                     WHERE w.e_id = new.e_id AND
                           w.e_id = d.d_id
                   );
  END;
```

3. The following is a schedule with one action missing:

T_1,	T_2
R(A)	
	R(B)
? ? ?	
W(C)	
	W(A)

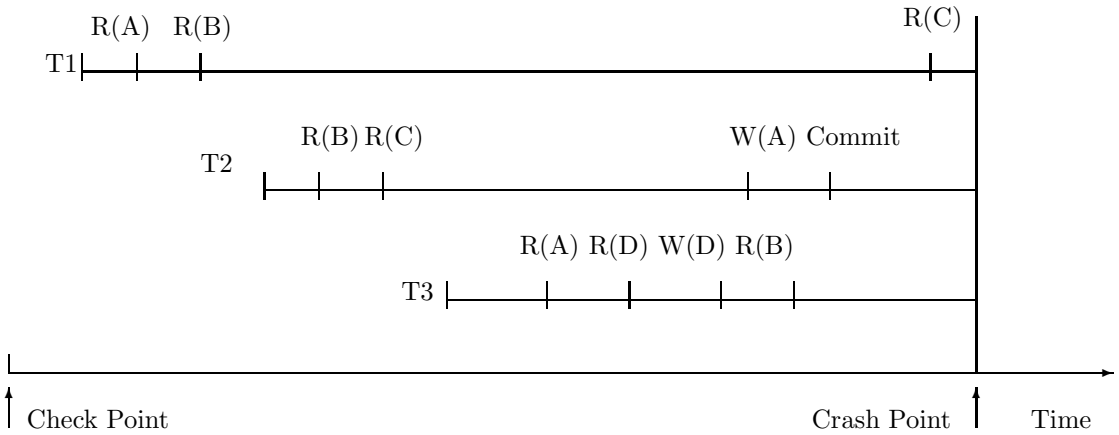
You are asked to figure out what actions could replace the ??? and make the schedule not serializable. List **all** possible non-serializable replacements.

Note that a possible replacement is an action of the form $R(T_i, Q)$ or $W(T_i, Q)$, where $i = 1, 2$, and Q is one of A, B, C, indicating that transaction T_i reads from (or, write onto) data item Q. A non-serializable replacement is a replacement $R(T_i, Q)$ (or $W(T_i, Q)$) such that the given schedule is not serializable when ??? is replaced by $R(T_i, Q)$ (or $W(T_i, Q)$).

Solution:

$W(T_1, B)$, $W(T_2, C)$, and $R(T_2, C)$ are all non-serializable replacements.

4. Consider the following log information that involves three transactions. Assume that the immediate update protocol with check-pointing is used for crash recovery. And note that the log file does not necessarily record all operations of a transaction.



(a) Is the schedule, as shown in the log, recoverable? Why?

(b) Present an algorithm (list steps that must be taken) for recovering (ignore the recoverability).

Solution:

(a) The schedule is recoverable because T2, the only committed transaction, does not read any item modified by an active transaction.

(b) After restoring the database from the disk, do the following two steps:

i. redo W(A) by T2, which represents all the updates of T2; and

ii. undo W(D) by T3, which represents the set of all updates of T1 and T2.

5. Consider the following three transactions:

```
T1: R(A), W(B), W(A)
T2: R(B), W(C), W(B),
T3: R(C), W(A), R(C), W(A)
```

Generate a schedule for the above transactions that satisfies the strict two phase locking protocol. The protocol requires that (1) if a transaction wants to read (respectively modify) an object, it first requests a shared lock (respectively exclusive) lock on the object, and (2) all locks held by a transaction are released only when the transaction is completed.

Your schedule should include not only the read and write operations but also the respected lock operations, and a transaction shall not request a lock unless it is necessary. A transaction shall be rolled back whenever the execution of a statement of it causes deadlock.

Schedule your transactions in a *round-robin fashion*, i.e., first transaction T1 is allowed to execute its first action, then T2, then T3, and after that T1 again, and so on – if that is possible according to the protocol. In addition, if a transaction is aborted, restart it again after the other transaction have executed their next operation (if possible).

Solution:

T1	T2	T3
s_lock(A)		
R(A)		
	s_lock(B)	
	R(B)	
		s_lock(C)
		R(C)
wait_for(e_lock(B))		
	wait_for(e_lock(C))	
		wait_for(e_lock(A)
		rollback
	e_lock(C)	
	W(C)	
		wait_for(s_lock(C))
	e_lock(B)	
	W(B)	
	release_lock(B,C)	
	commit	
		s_lock(C)
		R(C)
e_lock(B)		
W(B)		
		wait_for(e_lock(A))
e_lock(A)		
W(A)		
release_lock(A,B)		
commit		
		e_lock(A)
		W(A)
		s_lock(C)
		R(C)
		e_lock(A)
		W(A)
		release_lock(A,C)
		commit

6. Consider the following XML document, stored at *www.ourbookstore.com/books.xml*.

```
<?xml version="1.0" encoding="utf-8"?>
<book isbn="0836217462" format = "hardcover">
   <title>
        Being a Dog Is a Full-Time Job
   </title>
   <authors>
        <author  aid = "12345">
             <firstname> Charles </firstname>
             <lastname>  Schulz  </lastname>
             <email> schulz@hotmail.com </email>
```

```
        </author>
    </authors>
    <published>
        1997
    </published>
</book>
<book isbn="23650395454" format = "paperback">
    <title>
        The Character of Physical law
    </title>
    <authors>
        <author aid = "54321">
            <firstname> Richard </firstname>
            <lastname> Brewka </lastname>
            <email> brewka@www.com </email>
         </author>
         <author  aid = "12345">
            <firstname> Charles </firstname>
            <lastname>  Schulz  </lastname>
            <email> schulz@hotmail.com </email>
        </author>
    </authors>
    <published>
       1979
    </published>
</book>
```

Write an XQuery expression to list the title of all books written by Charles Schulz (i.e., the first name is Charles and last name Schulz).

Solution:

```
FOR #b IN doc(www.ourbookstore.com/books.xml)/book
WHERE $b/authors/author/firstname = 'Charles' AND
      $b/authors/author/lastname = 'Schultz'
RETURN
      <RESULT>  $b/title</RESULT>
```

7. Consider the above XML document again. For the following two XQuery expressions, present the result of each query in the form of XML documents.

   (a)
```
FOR $b IN doc(www.ourbookstore.com/books.xml)//author
RETURN <RESULT>
          <firstname>$b/firstname </firstname>
           <lasttname>$b/lastname </lastname>
       </RESULT>


<RESULT>
    <firstname><firstname> Charles </firstname></firstname>
    <lastname><lastname> Schulz </lastname></lastname>
```

```
</RESULT>
<RESULT>
    <firstname><firstname> Richard </firstname></firstname>
    <lastname><lastname> Brewka </lastname></lastname>
</RESULT>
<RESULT>
    <firstname><firstname> Charles </firstname></firstname>
    <lastname><lastname> Schulz </lastname></lastname>
</RESULT>
```

(b)

```
LET $b IN doc(www.ourbookstore.com/books.xml)//author
RETURN <RESULT>
          <AUTHOR>
            <firstname>$b/firstname </firstname>
            <lasttname>$b/lastname </lastname>
          </AUTHOR>
       </RESULT>

<RESULT>
  <AUTHOR>
     <firstname><firstname>Charles</firstname></firstname>
     <lastname><lastname>Schulz</lastname></lastname>
  </AUTHOR>
  <AUTHOR>
     <firstname><firstname>Richard</firstname></firstname>
     <lastname><lastname>Brewka</lastname></lastname>
  </AUTHOR>
  <AUTHOR>
     <firstname><firstname>Charles</firstname></firstname>
     <lastname><lastname>Schulz</lastname></lastname>
  </AUTHOR>
</RESULT>
```