

NOTE: PROBLEM 3 SOLUTION IS NOW INCLUDED IN THIS KEY!!!

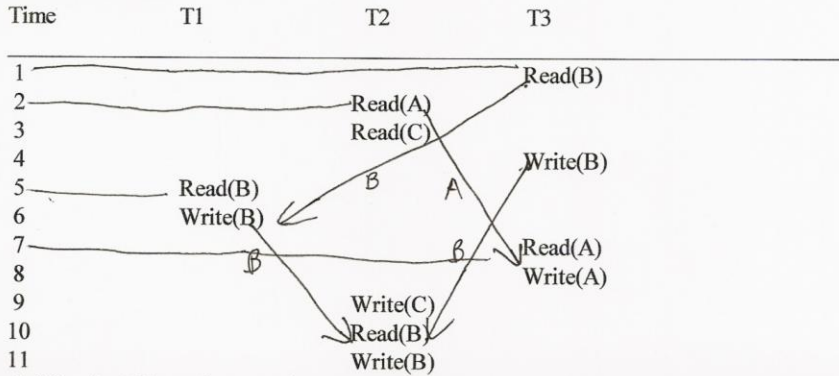
CINS 370 Sample Final Exam for Review

Dr. Melody Stapleton

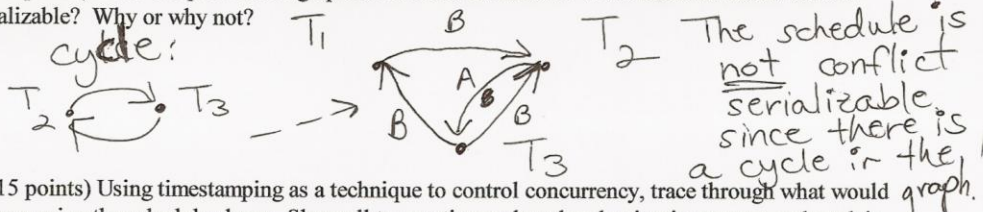
NAME _____

Directions: This test is open book and notes. Show your work on all problems, giving any (reasonable) assumptions. 100 total points. Points per problem are listed in parentheses in front of each problem. Five total pages.

1. (25 points total) Use the schedule below to answer the following subparts to this question. You may assume that at time 12 none of the transactions is complete yet, i.e. none have reached their end of transaction:



a) (10 points) Draw the precedence graph for the above schedule. Is the above schedule conflict-serializable? Why or why not?



b) (15 points) Using timestamping as a technique to control concurrency, trace through what would happen using the schedule above. Show all transaction and read and write timestamps and explain what would happen with each advance of time:

~~TS~~ $TS(T_1) = 5$ $TS(T_2) = 2$ $TS(T_3) = 1$

$\frac{ReadTS(A)}{\emptyset \times 5}$	$\frac{ReadTS(B)}{\emptyset \times 5}$	$\frac{ReadTS(C)}{\emptyset \times 2}$	At time $T=8$ T_3 issues $write(A)$ but $readTS(A) = 2 > 1$ so abort T_3 & rollback
$\frac{WriteTS(A)}{0}$	$\frac{WriteTS(B)}{\emptyset \times 5}$	$\frac{WriteTS(C)}{0}$	
$TS(T_3) = 1$			

at time 7 leave alone.

SQL Queries:

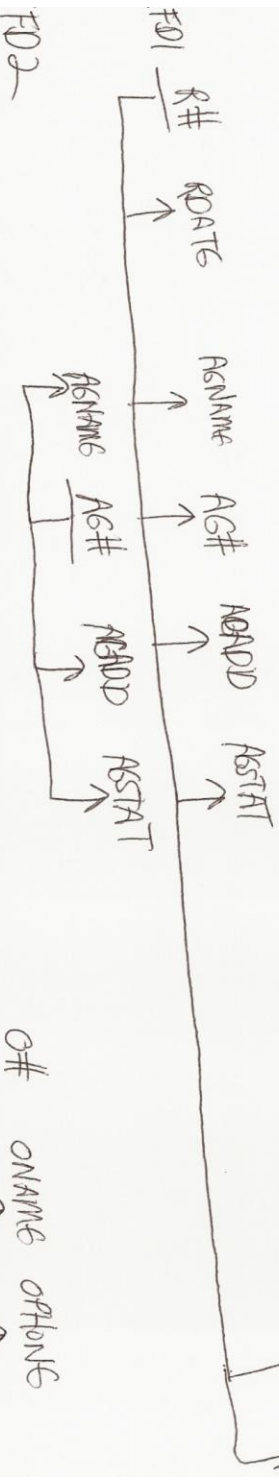
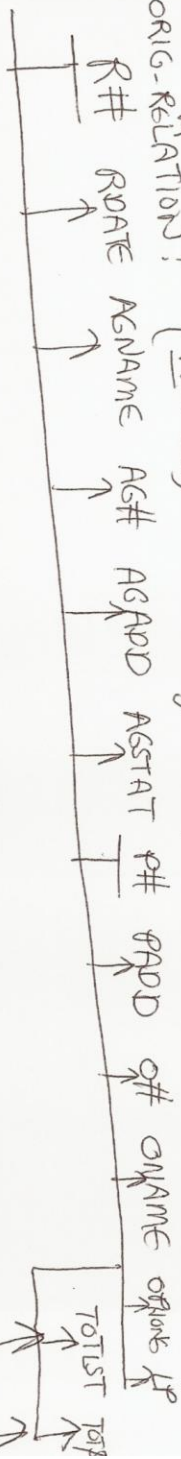
2a)

```
SELECT C.COURSENAME, S.SECTIONIDENTIFIER,  
       COUNT(G.STUDENTNUMBER)  
FROM COURSE C JOIN SECTION S JOIN  
      GRADEREPORT G  
WHERE S.SEMESTER = 'Fall' AND  
       S.YEAR = 2002  
GROUP BY C.COURSENAME, S.SECTIONIDENTIFIER;
```

2b) SELECT ^{DISTINCT} C.COURSENAME, C.COURSENUMBER
FROM COURSE C JOIN SECTION S
WHERE S.INSTRUCTOR = 'Hilzer' AND
 S.SEMESTER = 'Fall' AND S.YEAR = 1997
ORDER BY C.COURSENUMBER;

Problem 3: Assumptions: Report number uniquely determines the agent

ORIG-RELATION: (R#, P#) is the key



FD2

FD3

FD4

Assuming ^{List Price} and Owners of property change over time, here are our final, normalized relations

AGENT(A#, AGNAME, AGADD, AGSTAT)
 AGREPORT(R#, RDATE, AG#, TOTLIST, TOTP)
 PROPERTY(P#, PADD)
 OWNER(O#, ONAME, OPHONS)

4. (15 points) For the parallel schedule given below for transactions T1, T2 and T3 :Trace through the schedule below and show where deadlock occurs using a rigorous two-phase locking scheme. Assume that a locking scheme of upgradeable locks is used. I.e., when one wants to read a data item, they ask for a shared lock, when they want to write that same data item later on, they request an exclusive lock. Explain how the wait-die scheme for deadlock prevention would prevent deadlock from occurring.

time	T1	T2	T3
1		SLK read(A)	
2		SLK read(B)	
3	SLK read(X)		
4	SLK read(A)		
5		XLK write(B)	
6		write(A)	
7		wait	SLK read(X)
8		!	SLK read(A)
9	write(X)		
10	wait		read(B) - EOT
11	write(A) - EOT		wait
12		read(Y) - EOT	!

- At t=6 T2 requests XLOCK on A but since T1 has an SLOCK on A T2 is required to wait.
- At t=9 T1 requests XLOCK on X but since T3 has and SLOCK on X T1 is required to wait.
- At t=10 T3 requests an SLOCK on B but since T2 has an XLOCK T3 is required to wait

Deadlock occurs at time 10.

Wait-die: $TS(T_2) = 1, TS(T_1) = 3, TS(T_3) = 7$
 At time t=6 since T2 is older ^{than T1} it is allowed to wait.
 At time t=9 since T1 is older than T3 it is allowed to wait.
 At time t=10 since T3 is younger than T2, T3 is rolled back & restarted w/ the same time stamp.

5. (20 points) The following list represents the log entries for four transactions T1, T2, T3, and T4 at the point of a system crash. Suppose that the immediate update protocol with checkpointing has been used. Describe the recovery process from the point of the system crash. Describe how recovery occurs in this situation. Suppose that the initial values of variables are X=45, Y=65, A=70, B=25. What are the values of each of these variables after recovery takes place? Is this a recoverable schedule? Is there any cascading rollback? At the end of your recovery, in giving the values of variables, assume only undo and redo has occurred, but no fail transactions have been rerun as yet.

NOTE: The form of the write statements in the log is: [write_item, transaction_no, variable, old_value, new_value]

```
[start_transaction, T4]
[read_item, T4, A]
[start_transaction, T1]
[write_item, T4, A, 70, 75]
[read_item, T4, B]
[start_transaction, T3]
[read_item, T3, X]
[write_item, T3, X, 45, 55]
[start_transaction, T2]
[write_item, T4, B, 25, 30]
[read_item, T2, B]
[read_item, T1, X]
[read_item, T1, Y]
[commit T4]
[write_item, T2, B, 30, 92]
[checkpoint; L = T1, T2, T3]
[read_item, T2, A]
[write_item, T2, A, 75, 23]
[commit T2]
[read_item, T1, Y]
[write_item, T1, Y, 65, 82]
[write_item, T1, X, 55, 77]
[commit T3]
<----- system crash
```

X	Y	A	B
45	65	70	25
55	82	75	30
77	65	23	92
55			

- 1) Committed before checkpoint \downarrow T4: do nothing
- 2) Active List: T1, T2, T3
Redo list: T2, T3
Undo list: T1

a) After undo:

X	Y	A	B
55	65	23	92

b) values are same as above after redoing T2, T3

3) Schedule is recoverable since no committed transactions need to be rolled back.

4) No cascading rollback since no other transaction read values written by T1

More Practice! Here is another Immediate Update Problem:

The following list represents the log entries for four transactions: T1, T2, T3, and T4 at the point of a system crash. Suppose that the immediate update protocol with checkpointing has been used. Describe the recovery process from the point of the system crash. Describe how recovery occurs in this situation. Suppose the initial values of variables are X=30, Y=10, A=20, B=50. Is this a recoverable schedule? What are the values of each of these variables after recovery takes place? Is there a cascading rollback? At the end of your recovery, in giving the values of variables, assume

only undo and redo has occurred, but no failed transactions have been rerun as yet.

Note: the form of the write statements in the log is: [write_item, transaction_num, variable, old_value, new_value]

Start of log...
 [start_transaction, T2]
 [read_item, T2, X]
 [start_transaction, T3]
 [write_item, T2, X, 30, 40]
 [read_item, T2, Y]
 [read_item, T3, X]
 [write_item, T2, Y, 10, 20]
 [read_item, T3, Y]
 [commit, T2]
 [start_transaction, T1]
 [start_transaction, T4]
 [read_item, T1, A]
 [write_item, T1, A, 20, 30]
 → [checkpoint, L= {T4, T3, T1}]
 [read_item, T3, A]
 [read_item, T4, B]
 [write_item, T3, A, 30, 35]
 [write_item, T4, B, 50, 60]
 [commit, T4]
 [read_item, T1, B]
 [write_item, T1, B, 60, 70]
 <..... system crash

X	Y	A	B
30	10	20	50
40	20	30	60
		35	70
		30	60
		20	

1) Committed before checkpoint: T2
 so do nothing

2) Active List = T4, T3, T1
 Redo: T4
 Undo: T1, T3

a) After undo:

X	Y	A	B
40	20	20	60

b) Redoing T4: values already reflected above

3) Recoverable since no committed transaction need to be rolled back

4) ~~no cascading~~ cascading rollback ~~since~~ occurs since T3 read the value written by T1 for A

Return to [Melody's Home Page](#).