THE UNIVERSITY OF AUCKLAND

FIRST SEMESTER, 2013
Campus: CITY

Computer Science

TEST

Fundamentals of Database Systems

(Time Allowed: 50 minutes)

Note:

- The use of calculators is NOT permitted.
- Compare the exam version number on the Teleform sheet supplied with the version number above. If they do not match, ask the supervisor for a new sheet.
- Enter your name and student ID on the Teleform sheet. Your name should be entered left aligned. If your name is longer than the number of boxes provided, truncate it.
- Answer all **Multiple-choice** questions on the Teleform answer sheet provided. Answer Section **B** in the space provided in this booklet. Attempt all questions.
- Use a dark pencil to mark your answers in the multiple choice answer boxes on the Teleform sheet. Check that the question number on the sheet corresponds to the question number in this question/answer book. If you spoil your sheet, ask the supervisor for a replacement.
- Write your answers in the space provided in the short answer section. Write as clearly as possible. The space provided will generally be sufficient but is not necessarily an indication of the expected length. Extra space is provided at the end of this exam book.

Surname:	
First Name(s):	
Student ID:	
Login Name(UPI):	

MARKERS ONLY

Question	Marks	Out Of
18		4
19		8

Model Solutions for Test 1, Version 1

For the following four questions use the following set of transactions and the indicated timing of when the operations are issued.

TA1: r1[x], w1[x], c1
TA2: r2[x], w2[x], c2
TA3: r3[x], c3

Question 1
[2 marks] If we use the simple scheduler, which operation gets blocked first?

(a) w2[x],
(b) **r2[x],**
(c) r3[x],
(d) w1[x],
(e) None of the above

Question 2
[2 marks] If we use the common scheduler, which operation gets blocked first?

(a) **w1[x],**
(b) r3[x],
(c) w2[x],
(d) r2[x],
(e) None of the above

Question 3
[2 marks] If we run all transactions on the isolation level READ COMMITTED, i.e. they ignore each other's read locks, which operation gets blocked first?

(a) w1[x],
(b) r2[x],
(c) **r3[x],** (w1[x], ignores read lock)
(d) w2[x],
(e) None of the above

Question 4
[2 marks] If we run TA1, TA2 on the Isolation level READ COMMITTED, and TA3 on isolation level READ UNCOMMITTED, which operation gets blocked first?

(a) w1[x],
(b) r3[x],
(c) r2[x],
(d) **w2[x],** (additionally r3[x] ignores write lock)
(e) None of the above

For the following three questions use the following schedule
s1: r1[z], w2[x], r3[z], r1[x], w3[z], c3, r1[z], r1[y], w1[y], c1, c2

Question 5
[3 marks] What is the worst phenomenon in schedule s1?

(a) **Dirty write** (w1[y] is a dirty write, because r1[x] is a dirty read)
(b) Fuzzy read
(c) Lost update
(d) Dirty read
(e) None of the above

Question 6
[2 marks] Name the first object on which the locking rules of the common scheduler are not observed?

(a) z
(b) y
(c) **x** (this happens at the dirty read r1[x])
(d) None of the above.

Question 7
[2 marks] Which transaction is the first to not observe the locks on an object?

(a) **TA1**
(b) TA2
(c) TA3
(d) None of the above.

For the following three questions use the following schedule
s2: r1[z], w2[x], r3[z], c2, r1[x], w3[z], c3, r1[y], w1[y], c1

Question 8
[3 marks] What is the worst phenomenon in schedule s2?

(a) Lost update
(b) Fuzzy read
(c) Dirty read
(d) Dirty write
(e) **None of the above**

Question 9
[2 marks] Name the first object on which the locking rules of the common scheduler are not observed?

(a) **z** (at w3[z])
(b) y
(c) x
(d) None of the above

Question 10
[2 marks] Which transaction is the first to not observe the locks on an object?

(a) TA1
(b) TA2
(c) **TA3**
(d) None of the above

For the following three questions use the following set of transactions. We use the **common** scheduler.
TA1: r1[x], w1[x], w1[y], r1[y], c1 Deadlock: r1[x], w1[x]___?,
TA2: r2[y], r2[k], r2[z], c2
TA3: r3[z], w3[x], r3[x], c3 w3[x]_____?,

Question 11
[3 marks] Can **TA1** become part of a deadlock, and if yes, which operation would be blocked?

(a) Yes, w1[y] would be blocked.
(b) Yes, r1[x] would be blocked.
(c) **Yes, w1[x] would be blocked.**
(d) Yes, r1[y] would be blocked.
(e) None of the above

Question 12
[2 marks] Can **TA2** become part of a deadlock, and if yes, which operation would be blocked?

(a) Yes, r2[z] would be blocked.
(b) Yes, r2[k] would be blocked.
(c) Yes, r2[y] would be blocked.
(d) **None of the above** (only r2[y] could be blocked but not be part of a deadlock)

Question 13
[2 marks] Can **TA3** become part of a deadlock, and if yes, which operation would be blocked?

(a) **Yes, w3[x] would be blocked.**
(b) Yes, r3[x] would be blocked.
(c) Yes, r3[z] would be blocked.
(d) None of the above

For the following three questions, consider the following scenario:
The stable database has at time t the following pages, objects on these pages and values:

Page 1:
x = 54
y = 37
Page 2:
z = 23
k = 95

The following is the list of the stable log records at time t. The database uses the steal, no-force policy. The database encounters a system crash at time t. You are supposed to **perform crash recovery**.

[nr: 161, ta: 81, obj: z, b: 12, a: 23]
[nr: 162, ta: 82, obj: k, b: 87, a: 95]
[nr: 163, ta: 82, obj: x, b: 52, a: 54]
[nr: 164, ta: 82, commit]
[nr: 165, ta: 83, obj: y, b: 61, a: 37]
[nr: 166, ta: 81, obj: x, b: 54, a: 78]
[nr: 167, ta: 81, commit]
[nr: 168, ta: 84, obj: x, b: 78, a: 21]
[nr: 169, ta: 84, obj: z, b: 23, a: 34]
[nr: 170, checkpt: redo: 161, undo: 165]
[nr: 171, ta: 83, obj: k, b: 95, a: 73]

Question 14
[2 marks] What is the content of page 1 of the stable database after the crash recovery?

(a) x= 54, y=37
(b) x= 78, y=37
(c) x= 52, y=61
(d) **x= 78, y=61**
(e) None of the above

Question 15
[2 marks] What is the content of page 2 of the stable database after the crash recovery?

(a) z= 12, k=87
(b) z= 12, k=95
(c) **z= 23, k=95**
(d) z= 34, k=73
(e) None of the above

Question 16
[2 marks] Was a database buffer page written to the stable database applying the steal policy, i.e. containing a then uncommitted write? If Yes, for which page and between which two log entries was this page written to the stable database?

(a) Yes, Page2 between log entry 169 and 170
(b) Yes, Page1 between log entry 163 and 164
(c) Yes, Page2 between log entry 162 and 163
(d) **Yes, Page1 between log entry 165 and 166** (only there x and y have the values needed)
(e) None of the above

Question 17
[3 marks] Consider the following set of transactions, with the indicated timing of when the operations are issued. They will run into a deadlock in the common scheduler.
TA1: w1[y], R1[x], w1[x], c1
TA2: r2[x], w2[x], c2
What is the best strategy for resolving the deadlock:

(a) Abort TA1
(b) **Grant TA2 the exclusive lock on x, TA2 can continue, TA1 waits for TA2**
(c) Abort TA2
(d) Grant TA1 the exclusive lock on x; TA1 can continue, TA2 waits for TA1
(e) None of the above

SECTION B

Answer all questions in this section in the space provided. If you run out of space then please use the Overflow Sheet and indicate in the allotted space that you have used the Overflow Sheet.

Question 18 [4 marks]

Give reasons for your answer to Question 17

TA 1 has not yet seen x, so if TA2 is granted the exclusive lock and can finish, and then TA1 finishes,
then the resulting schedule is serializable:
s: w1[y], r2[x], w2[x], c2 R1[x], w1[x], c1
This is better than aborting one transaction.
Aborting any transaction would work, but would be unnecessary.
Granting TA1 the exclusive lock and then finishing TA2 would lead to a lost update.

Question 19 [8 marks]

a) Explain which operations should be performed in a transactional dequeue, how should the transaction demarcation be set?

In a transactional dequeue, one message is dequeued, e.g. marked as processed, and the appropriate operation is performed.
Both operations have to be within the same transaction, i.e. the Begin-of-Transaction has to be before that and the commit after that. Otherwise the intended effect is not achieved.

b) What does the transactional dequeue pattern achieve? What does it prevent?

The transactional dequeue achieves that a message is dequeued if and only if the appropriate action has been completely performed.
It prevents in particular that the message is dropped, e.g. marked as processed without the operation being performed. It also prevents the operation from being executed twice.

Overflow pages – please number answers carefully

Rough Working – This page will not be marked