Fachbereich für Computerwissenschaften

Prof. Dr. Nikolaus Augsten
Jakob-Haringer-Str. 2
5020 Salzburg, Austria
Telefon: +43 662 8044 6347
E-Mail: nikolaus.augsten@sbg.ac.at

Datenbanken Vertiefung — Prüfung
Wintersemester 2018/2019 — 06.02.2019

Name: ______________________ **Matrikelnummer:** ______________

Hinweise

- Bitte überprüfen Sie die Vollständigkeit des Prüfungsbogens (10 nummerierte Seiten).
- Schreiben Sie Ihren Namen und Ihre Matrikelnummer auf jedes Blatt des Prüfungsbogens und geben Sie alle Blätter ab.
- Grundsätzlich sollten Sie alle Antworten auf den Prüfungsbogen schreiben.
- Sollten Sie mehr Platz für eine Antwort benötigen, bitte einen klaren Verweis neben die Frage auf die Seitennummer des zusätzlichen Blattes setzen.
- Keinen Bleistift verwenden. Keinen roten Stift verwenden.
- Verwenden Sie die Notation und die Lösungsansätze, die während der Vorlesung besprochen wurden.
- Aufgaben mit mehr als einer Lösung werden nicht bewertet.
- Als Unterlage ist ein beliebig (auch beidseitig) beschriftetes A4-Blatt erlaubt.
- Zeit für die Prüfung: 90 Minuten

Unterschrift ______________________

Korrekturabschnitt — Bitte frei lassen

Aufgabe	1	2	3	4	5	6	7	8	Summe
Maximale Punkte	1	1	1	1	1	1	1	1	8
Erreichte Punkte									

Exercise 1 1 Point

Identify which of the properties: **conflict serializable**, **recoverable**, **cascadeless**, are fulfilled by the following schedule. If a property is not fulfilled, explain why.

T1:	T2:	T3:	T4:
read(A)			
			write(B)
		read(C)	
	read(B)		
	write(A)		
			COMMIT
	COMMIT		
		read(A)	
read(B)			
write(B)			
		write(A)	
		read(B)	
COMMIT			
		COMMIT	

Exercise 2 1 Point

Consider the following schedule. State if it is **conflict serializable** and explain why. Give an equivalent serial schedule if possible.

T1:	T2:	T3:	T4:
read(X)			
	write(X)		
		read(X)	
read(Y)			
	write(Y)		
read(Z)			
		write(Z)	
			read(Z)
			write(Z)

Exercise 3

1 Point

Consider the following schedule and the **two-phase locking** scheduler.

T1:	T2:	T3:
`read(X)`		
	`write(X)`	
		`read(X)`
`write(X)`		
		`write(X)`

Does the schedule result in a deadlock? If yes, what would happen to the transactions according to the wound-wait deadlock prevention protocol (assume $\mathrm{TS}(\mathrm{T1}) < \mathrm{TS}(\mathrm{T2}) < \mathrm{TS}(\mathrm{T3})$)?

Exercise 4 1 Point

Can the following schedule be the output of a strict two-phase locking scheduler? If yes, show the schedule with all required lock and unlock instructions. Otherwise, explain why.

T1:	T2:	T3:
read(X)		
read(Y)		
		read(X)
		write(X)
	read(X)	
read(Z)		
write(Z)		
COMMIT		
	write(Y)	
	COMMIT	
		read(V)
		COMMIT

Exercise 5

1 Point

Consider the following schedule and the **validation based** scheduler. The timestamps correspond to operation order.

T1:	T2:	T3:	T4:
start			
	start		
	validate		
			start
		start	
	finish		
validate			
finish			
		validate	
		finish	
			validate
			finish

The objects in the database that can be read or written are: A, B, C, D, E, F.

The read and write sets of the transactions are:

T1:	R-set(T1)={A,B},	W-set(T1)={C,D}
T2:	R-set(T2)={A,C},	W-set(T2)={D,F}
T3:	R-set(T3)={C,E},	W-set(T3)={B,F}

Answer the following questions.

1. Does T1 successfully validate?

2. Suppose that the validation of T4 and all previous validations succeeded. What could have been the largest read set of T4?

Exercise 6 — 1 Point

With the initial values A=100, B=200, C=300, write the log file for the following schedule. What happens during the recovery according to the recovery algorithm? Specify the resulting compensation log records.

T1:	T2:	T3:
	START	
START		
		START
		read(C)
	read(B)	
read(A)		
A:=A-20		
		C:=C+40
		write(C)
	B:=B-80	
write(A)		
	write(B)	
COMMIT		

----------CRASH---------------

Exercise 7 1 Point

Consider the following schedule. Indicate what happens when the schedule is processed by a **multiversion timestamp-ordering** scheduler. The transactions start in order with TS(T1)=1, TS(T2)=2, TS(T3)=3, TS(T4)=4. Assume that no versions of data item A exist.

T1:	T2:	T3:	T4:
write(A)			
			write(A)
		read(A)	
	write(A)		

Exercise 8 1 Point

Why does the following schedule **NOT** adhere to the **timestamp-ordering** protocol?

TS(T1)=1, TS(T2)=2, TS(T3)=3

T1:	T2:	T3:
	`read(A)`	
	`read(B)`	
	`write(B)`	
		`read(B)`
		`read(A)`
`read(C)`		
`write(C)`		
		`write(B)`
		`write(A)`
	`read(C)`	
`read(B)`		
`write(B)`		
	`write(C)`	