Fachbereich Informatik

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

PARIS LODRON UNIVERSITÄT SALZBURG

Datenbanken Vertiefung — Prüfung
Wintersemester 2022/2023 — 16.03.2023

Name: _____ **Matrikelnummer:** _____

Hinweise

- Bitte überprüfen Sie die Vollständigkeit des Prüfungsbogens (9 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

Consider the following schedule.

(a) Which transaction has to abort to trigger a **cascading rollback**? Why?

(b) Insert `COMMIT` instructions such that the schedule becomes cascadeless.

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

Exercise 2 — 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 3 — 1 Point

Can the following schedule be the output of a **two-phase** locking scheduler? If so, show the schedule with all required lock and unlock instructions. Otherwise explain why. Could it be the output of a ***strict* two-phase** locking scheduler? Why (not)?

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

Exercise 4 — 1 Point

Consider the following **multiversion two-phase locking** schedule. Assume that the starting value of `ts-counter` and the initial versions of data items A and B are 0. T_i and T_j are read-only transactions and start with their first read operation.

For each row in the schedule, indicate any

- changes to locks,
- waiting transactions,
- newly created data items,
- changes to versions of data items,
- and the data item version for reads.

Also, provide

- the timestamps of the read-only transactions
- and the final value of `ts-counter`.

Ti:	Tj:	Tk:	Tl:
		write(A)	
	read(A)		
			read(A)
	read(B)		
		commit	
			write(B)
read(A)			
			commit
read(B)			

Exercise 5 — 1 Point

Consider the **snapshot isolation** concurrency control scheme and the following transactions. Explain the *first committer wins* and *first updater wins* rules in this scenario, can both transactions commit? Why (not)? Without any rule, which anomaly would occur in this case?

T1:	T2:
read(A)	
	read(A)
read(B)	
write(B)	
	read(B)
	write(B)
	COMMIT
COMMIT	

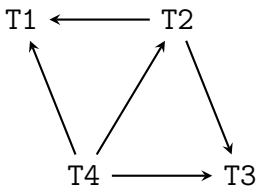
Exercise 6 — 1 Point

Consider the following graph in which a directed edge `Ti`→`Tj` indicates that `Ti` requests a lock currently held by `Tj`.

(a) Which of the transactions would be rolled back under the **wound-wait** deadlock prevention strategy?

(b) With what timestamp is the rolled-back transaction restarted and why?

Transaction timestamps are equal to their numeric identifiers (smaller numbers are older).



Exercise 7 — 1 Point

With the following starting values:

A=10, B=20, C=30, D=40, E=50, F=60

write the log file (physical logging) for the following schedule including the log records generated during recovery.

```
T1:         T2:         T3:
start
read(A)
A:=A-5
write(A)
            start
            read(C)
read(B)
B:=B-5
write(B)
read(D)
            C:=C+15
                        start
                        read(E)
            write(C)
            COMMIT
----------CHECKPOINT------------
D:=D-10
write(D)
COMMIT
                        E:=E-15
                        write(E)
                        read(F)
                        F:=F-15
------------CRASH---------------
```

Exercise 8 — 1 Point

Consider a multi-granularity locking protocol with lock modes S, X, IS, IX, SIX. The following object hierarchy is given:

- root R with blocks A and B,
- block A contains records a1 and a2,
- block B contains records b1, b2, and b3.

Write the list of locks (objects being locked and lock type) required to execute the following transactions. Start with the root.

T1: modify record a2

T2: read all records in block B

Can T1 and T2 be executed concurrently? Why?