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

PARIS LODRON UNIVERSITÄT SALZBURG

Datenbanken Vertiefung — Prüfung
Wintersemester 2021/2022 — 19.01.2022

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

Is the following schedule **conflict serializable**? If it is, give an equivalent serial schedule. If it is not, explain why.

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

Exercise 2 — 1 Point

Consider the following schedule.

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

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

(c) Why is cascadelessness of a schedule desirable?

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

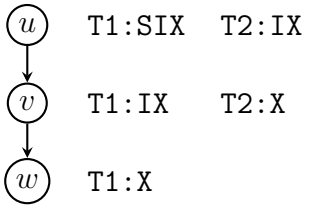
Exercise 3 — 1 Point

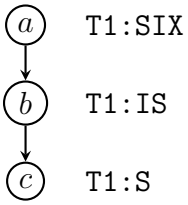
Consider the following schedule and a *strict* two-phase locking scheduler *with* lock conversions. Insert lock, unlock, upgrade, and commit instructions such that all read and write operations in the schedule can be performed in this order.

```
T1:       T2:       T3:       T4:
_____________________________________

          read(A)
          write(A)



                                read(B)

read(B)

                    read(A)



                                write(B)


read(A)
write(A)

_____________________________________
```

Exercise 4 — 1 Point

Consider the **multiple granularity** locking scheme with intentional locks. The following two figures show data hierarchies. At each node, there is a transaction and a lock type it holds. For each figure, identify which rules of the scheme are violated or state that all rules are obeyed. Can transactions `T1` and `T2` run concurrently? Justify why (not).





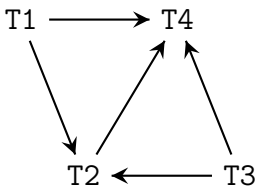
Exercise 5 — 1 Point

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 **wait-die** 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 6 — 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 7 — 1 Point

Consider the following schedule. Indicate what happens at each step 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)		
			read(A)
read(A)			
		write(A)	

Exercise 8 — 1 Point

With the initial values A=50, B=100, C=150, D=200, write the log file (physical logging) for the following schedule including the log records generated during recovery.

```
T1:          T2:          T3:
start
read(A)
                          start
                          read(C)
A:=A+50
write(A)
                          C:=C-100
COMMIT
----------CHECKPOINT------------
             start
             read(B)
             read(D)
                          write(C)
             D:=D-B
             write(D)
------------CRASH---------------
```