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 2023/24 — 24.05.2024

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									

Aufgabe 1 — 1 Punkt

Is the following schedule **conflict serializable**? Draw a precedence graph to verify. If it is not, explain why. If it is, give an equivalent serial schedule.

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

Aufgabe 2 — 1 Punkt

Consider the following schedule.

(a) Which transaction must abort to trigger a **cascading rollback** of all other transactions?

(b) Show the commit order (by inserting the commit commands into the schedule) such that the schedule is **cascadeless**.

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

Aufgabe 3 — 1 Punkt

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

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

                        read(A)
                        write(A)
            write(B)
            COMMIT

                        read(A)
                        read(B)
read(B)
read(A)
                        COMMIT
COMMIT
──────────────────────────────
```

Aufgabe 4 — 1 Punkt

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	

Aufgabe 5 — 1 Punkt

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 the first write of data item `A` succeeds. State if a transaction must be rolled back and also consider cascading rollbacks.

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

Aufgabe 6 — 1 Punkt

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

(a) Assume that all transactions want to issue a `write` operation on item `A` in the following order: `T1`, `T2`, `T3`. Complete the schedule by inserting lock requests (e.g., `R:lock-S(A)`) and granted locks (e.g., `G:lock-S(A)`) for all remaining `read` and `write` operations. Denote in the schedule if a transaction has to wait since another transaction holds an incompatible lock.

(b) Draw the wait-for graph. Does the schedule result in a deadlock? Why?

(c) Briefly explain the *wound-wait* deadlock prevention scheme on this example by assuming `TS(T1)` $<$ `TS(T2)` $<$ `TS(T3)`.

```
T1:         T2:         T3:
_____________________________________
R:lock-S(A)
G:lock-S(A)
read(A)
_____________________________________

                        read(A)
_____________________________________

_____________________________________

_____________________________________

_____________________________________
```

Aufgabe 7 — 1 Punkt

Consider the following log.

1. Which event has triggered the log record in line 10?
2. What would happen during the recovery if the system crashes at the end of the log? Indicate the generated log records.

```
 1 <T1, start>
 2 <T1, D, 10, 20>
 3 <T1, commit>
 4 <T4, start>
 5 <T4, A, 0, 20>
 6 <checkpoint, {T4}>
 7 <T2, start>
 8 <T2, C, 15, 25>
 9 <T3, start>
10 <T4, A, 0>
11 <T3, B, 20, 30>
12 <T2, commit>
-------CRASH---------
```

Aufgabe 8 — 1 Punkt

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).