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 — 25.01.2023

**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

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

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

## Exercise 2 — 1 Point

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) | |

## Exercise 3 — 1 Point

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
_____________________________
```

## Exercise 4 — 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. 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) | | |

## Exercise 5 — 1 Point

Consider the following schedule and the **validation based** scheduler. The timestamps correspond to the validation order, i.e., `TS(Ti)=validation(Ti)`.

| T1: | T2: | T3: | T4: |
|---|---|---|---|
| | start | | |
| start | | | |
| validate | | | |
| finish | | | |
| | | | start |
| | | start | |
| | 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,C}, W-set(T1)={D,F}
T2: R-set(T2)={A,B}, W-set(T2)={B,D}
T3: R-set(T3)={C,E}, W-set(T3)={B,F}
T4: R-set(T4)={E,F}, W-set(T4)={B,E}

Which of the four transactions can not be committed? State which condition is violated.

## Exercise 6 — 1 Point

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: `T3`, `T2`, `T1`. 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 *wait-die* 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)
_____________________________________

_____________________________________

_____________________________________

_____________________________________
```

## Exercise 7 — 1 Point

Consider the following log.

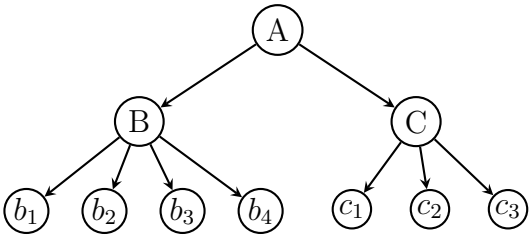
(a) What would happen during recovery if the system crashes at the end of the log? Indicate the generated log records.

(b) After the redo phase, which transactions are contained in the undo list? What is the purpose of the redo phase?

```
 1: <T1, start>
 2: <T1, D, 10, 20>
 3: <T3, start>
 4: <T3, A, 20, 10>
 5: <T1, commit>
 6: <T3, B, 30, 45>
 7: <checkpoint, {T3}>
 8: <T2, start>
 9: <T2, C, 10, 25>
10: <T4, start>
11: <T2, E, 0, 10>
12: <T4, D, 20, 15>
13: <T4, commit>
-------CRASH---------
```

## Exercise 8 — 1 Point

Consider the **multiple granularity** locking scheme with intentional locks. The following figure shows a data hierarchy: for instance, the root level A is a database, B and C are relations, and $b_i, c_i$ are data items.




Now, consider the following tasks that the transactions should execute.

- `T1`: Update a value of data item $c_3$.
- `T2`: Read all data items contained in B such that it is possible to update a few of them that satisfy a specific condition.
- `T3`: Read data items $b_1$ and $c_1$.

(a) Indicate the locks acquired by each of the transactions (assume that no locks are released).

(b) Which of the transactions can run concurrently, and why?