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 — 31.01.2024

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

Answer the following questions (be concise):

1. Explain three *undesirable phenomena* of concurrent transactions.
2. What does *read uncommitted* stand for?
3. What is a *lost update* and how can it be prevented?

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 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)
--------------------------------------
```

Aufgabe 5 — 1 Punkt

Consider the **snapshot isolation** concurrency control scheme and the following transactions.

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

(a) Can `T1` and `T2` both commit? Explain why.

(b) What would the use of `select ... for update` for all read operations change in this scenario?

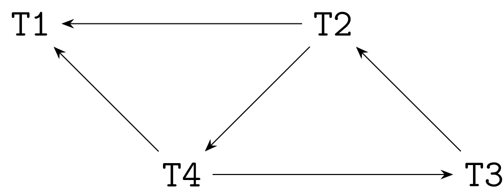
Aufgabe 6 — 1 Punkt

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. Older transactions have smaller timestamps.



Aufgabe 7 — 1 Punkt

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---------------
```

Aufgabe 8 1 Punkt

Is the following schedule valid under the **timestamp-ordering** protocol? If yes, show the timestamps for the affected data items for each operation. If not, mark the first problematic operation and explain the problem.

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(B)	
		write(A)	
write(C)			
	read(A)		