Department of Computer Science

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

PARIS LODRON UNIVERSITÄT SALZBURG

Advanced Databases
Winter Semester 2024/25

Exam
29.01.2025

Name: _____ **Student ID:** _____

Hints

- Check whether you received all pages of the exam (9 pages).
- Write your name or your student ID on each sheet of the exam and hand in all pages.
- All answers are expected to be written on the exam sheets.
- Clearly highlight and enumerate additional pages that are used for longer answers. Match your text with the according exercise.
- Only use pencils that are permanent and non-red colored.
- Use the notation and techniques discussed in the lecture.
- Exercises with more than one solution are not graded.
- You are allowed to use one A4 sheet with your personal notes (both sides, hand written or printed).
- Exam duration: 90 minutes

Signature _____

Grading — Filled by the examiner

Exercise	1	2	3	4	5	6	7	8	Sum
Total points	1	1	1	1	1	1	1	1	8
Points reached									

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 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. **Avoid unlock and commit operations that are not necessary.**

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

           read(A)
           write(A)



                                  read(B)

read(B)

                      read(A)



                                  write(B)


read(A)
write(A)

_____________________________________
```

Exercise 3 — 1 Point

Consider the following schedule and the **validation based** scheduler.

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

Answer the following questions.

1. Can `T2` be committed? Explain which conditions are satisfied or which one is violated.
2. Suppose that the validation of T4 and all previous validations succeeded. What could have been the largest read set of T4?

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:
		read(A)	
		write(A)	
read(A)			
			read(A)
		commit	
	read(B)		
	read(A)		
			write(B)
			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)
	write(A)
read(B)	
write(A)	
	write(B)
COMMIT	
	COMMIT

Exercise 6 — 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: read all records in block A

T2: modify record b3

Can T1 and T2 be executed concurrently? Why?

Exercise 7 — 1 Point

Prove that *lock ordering* (using a total order) prevents deadlocks.

Reminder: In *lock ordering*, all data items are ordered by some total order. Once a transaction has locked a particular item, it cannot request locks on items that precede that item.

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