Department of Computer Science

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

Non-Standard Database Systems — Exam
Summer Semester 2022/2023 — 27.06.2023

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 sheet.
- Number any additional pages that you use for longer answers and reference them clearly visible on the corresponding exercise sheet.
- Use only pencils that are permanent. No red pencil.
- Use the notation and techniques discussed during the lecture.
- Exercises with more than one solution are not graded.
- You may use one A4 sheet with your personal notes (on 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	Summe
Total Points	2	2	2	2	2	2	2	2	16
Points Reached									

Exercise 1 - *Throughput and Response Time.* 2 Points

A database system must process transactions t_1, t_2, t_3, and t_4. For each transaction, Table 1 shows the start time (when the transaction enters the system) and the execution time (the runtime of the transaction in the system when it is not interrupted).

Compute the average response time and the throughput for the following task scheduling strategies where shorter transactions are given priority over longer transactions (i.e., the waiting queue is ordered by execution time):

1. No running transaction is aborted.
2. Abort the running transaction when a shorter transaction enters the waiting queue.

Transaction	Start Time	Execution Time
t_1	0	3
t_2	1	4
t_3	2	2
t_4	3	1

Table 1: Start and execution time of transactions.

Exercise 2 - *Horizontal Partitioning*. 2 Points

Relation $r[A]$ in Table 2 should be horizontally partitioned onto four disks, D_i, $0 \leq i \leq 3$. Partition the tuples on attribute A using the following partitioning strategies:

1. hash partitioning,
2. range partitioning.

Propose an appropriate hash function/partitioning vector that avoids data skew.

A	30	72	54	46	66	34	42	60	10	22	84	96

Table 2: Relation $r[A]$.

Exercise 3 - *Commit Protocols.* 2 Points

Mark the following statements as true (**T**) or false (**F**).

1. 2-Phase Commit does not block as long as the coordinator is reachable. ☐
2. 3-Phase Commit may block when $k + 2$ sites fail at the same time. ☐
3. The first phase in 2-Phase Commit is identical to the first phase in 3-Phase Commit. ☐
4. The persistent messaging protocol avoids the receiver relation *received_messages* to grow too large by removing all messages that are older than a user-defined timeout. ☐

Exercise 4 - *Persistent Messaging.* 2 Points

Consider a sender S that sends a message to receiver R using the persistent messaging protocol. Table 3 shows the initial entries in the relations *messages_to_send* of the sender and *received_messages* of the receiver. Newer events have larger time stamps.

1. Compute the value of T_{OLD} for the relations in Table 3.
2. Show the relation *received_messages* after receiver R has received and processes the value of T_{OLD}.

messages_to_send

number	message	time	ack
1	$Q \leftarrow Q + 9$	2	received
3	$A \leftarrow A + 3$	5	received
7	$Q \leftarrow Q + 3$	6	
8	$B \leftarrow B - 9$	7	received
9	$C \leftarrow C - 6$	8	

received_messages

number	message	time	ack
1	$Q \leftarrow Q + 9$	2	sent
3	$A \leftarrow A + 3$	5	sent
7	$Q \leftarrow Q + 3$	6	sent
8	$B \leftarrow B - 9$	7	sent

Table 3: Relations *messages_to_send* at sender S and *received_messages* at receiver R.

Exercise 5 - *2-Phase Locking*. 2 Points

Execute the schedule in Figure 1 using rigorous 2-phase locking (i.e., a transaction releases its locks after the commit) with the majority-based locking protocol. Assume a distributed system with five nodes and full replication.

List the locking actions for each command in the schedule.

T_1	T_2	T_3
start		
R(A)		
		start
		R(A)
W(B)		
commit		
		W(B)
	start	
		R(B)
		commit
	W(A)	
	commit	

Figure 1: Transaction Schedule.

Exercise 6 - *Deadlock Handling*. 2 Points

Consider three transactions $T_1 : W(X), W(Y), W(Z)$, $T_2 : W(Y), W(Z), W(X)$, and $T_3 : W(Z), W(X), W(Y)$ that update data items X, Y, and Z.

1. Provide a schedule using 2-phase locking that results in a cycle that includes all transactions.
2. Draw the according local and the global wait-for graphs assuming a distributed system and a distributed lock manager. Data item X is managed by site S_1, Y is managed by site S_2, and Z is managed by site S_3.
3. Explain how the cycle is resolved in the centralized approach.

Exercise 7 - *Vector clocks.* 2 Points

Assume a single data item Q that is replicated on sites S_1, S_2, and S_3. A site S_i can do (i) a local write on Q, $W(Q)$, which changes the value of the local copy Q_i, or (ii) copy the value from a different site S_j, $C(S_j)$, $j \neq i$, which copies the value of Q_j to Q_i. All vectors are initialized with the zero vector.

1. Show the vector clocks resulting from the schedule in Figure 2.
2. Identify the first operation that leads to a branching history.

Note: Local reads, which will typically precede a local write in a real schedule, are not relevant for conflict detection and omitted from the schedule.

S_1	S_2	S_3
$W(Q)$		
		$W(Q)$
	$C(Q_3)$	
	$C(Q_1)$	
	$W(Q)$	
$C(Q_2)$		
		$C(Q_1)$

Figure 2: Schedule on replicated data item Q.

Exercise 8 - *Parallel Join.* 2 Points

Given a system with 16 processing nodes p_i, $1 \leq i \leq 16$, and two relations $R[A] = [4, 7, 13, 14, 16, 24, 25, 36, 44, 55, 57, 62, 68, 72, 78, 81]$ and $S[A] = [2, 7, 15, 21, 26, 32, 33, 36, 39, 41, 42, 53, 55, 56, 63, 78]$ (each number is an attribute value of A and forms a unary tuple). The relations are stored on nodes different from the processing nodes p_i.

Find an appropriate parallel join technique for the following query and count:

1. the number of tuples that must be transferred over the network between any pair of nodes;
2. the number of tuples from R and S that must be joined per processing node.

```
SELECT * FROM R, S
WHERE R.A = S.A;
```