Please fill in your name and registration number (Matrikelnr.) **immediately**.

EXAM IN	**SAMPLE SOLUTION**	21.02.2023 **GROUP A**
Matrikelnr.	Last Name	First Name

Duration: 90 minutes. Provide the solutions at the designated pages; solutions on additional sheets of paper are not considered. **Good Luck!**

Task	1	2	3	4	5	6	Σ
Max. Points	12	12	10	12	12	12	70
Points							

Please, *do not* remove the staple.

Add your student ID and last name on every sheet, it simplifies entering points.

Attention!

To all questions with a multiple choice option, the following rule applies: Just checking an option gives no points; points are only granted in combination with the required justification/example/...

Notation:

In Questions 1–3, the notation as known from the lecture slides and exercises for transactions T_i is used. Recall:

- $r_i(O)$ and $w_i(O)$: Read, respectively write operation of transaction T_i on object O.
- b_i, c_i, a_i: begin (`BEGIN OF TRANSACTION`), commit (`COMMIT`) and abort (`ABORT`/`ROLLBACK`) of T_i.

The indices $_i$ can be omitted if it is clear which transaction an operation belongs to.

In addition, log records also have the same format as used throughout the lecture:

`[LSN, TA, PageID, Redo, Undo, PrevLSN]` for "normal" records,
`[LSN, TA, BOT, PrevLSN]` for `BOT` log-records, and
`[LSN, TA, COMMIT, PrevLSN]` for `COMMIT` records.

Compensation log records (CLRs) follow the format
⟨`LSN, TA, PageID, Redo, PrevLSN, UndoNextLSN`⟩ and
⟨`LSN, TA, BOT, PrevLSN`⟩

In these records, `LSN` denotes the Log-Sequence Number, `TA` the transaction, `PageID` the page that was updated, `Redo` and `Undo` the information needed for the Redo resp. Undo operations, `UndoNextLSN` the LSN of the next log record of the same transaction to be undone, and `PrevLSN` the LSN of the previous log record of the same transaction.

In case of logical logging, the changes to the previous value of the database instance are stated only using *addition* and *subtraction*, e.g. [·, ·, ·, X+=d_1, X-=d_2, ·].

Question 1: Properties of transactions (12)

Assume that a DBMS implements the following isolation-levels, and each transaction is free to choose in which isolation-level it executes.

L1: Share-Locks are requested immediately before- and released immediately after each read operation (no 2PL); 2PL for Exclusive-Locks (independent of the releases of Share-Locks).

L2: combined 2PL for Share- and Exclusive-Locks, for Exclusive-Locks strict 2PL is used. Lock-Upgrades (of a Share-Lock to an Exclusive-Lock) are possible and do not count as a lock release.

Please answer the following questions with "yes" or "no". When answering *"yes"*, state a schedule (operations b_i, c_i, $r_i(O)$, $w_i(O)$) that is consistent with the stated isolation-level and satisfies the given property. When answering *"no"*, provide a short (1-2 sentences) justification.

Note: You need not state lock requests and releases of locks – just make sure that for your schedule a corresponding valid such sequence exists.

a) Consider the transaction

$$T_1 \colon b_1, r_1(A), w_1(A), w_1(B), r_1(A), r_1(B), r_1(A), c_1$$

and an arbitrary transaction T_2. Assume that T_1 runs in level L1 and T_2 in level L2.

i) Is it possible that a *Lost-Update* happens in T_2?

Lost-Update in T_2: ⊗ yes ○ no

`Possible solution:` .....

$b_1, b_2, r_1(A), r_2(A), w_2(A), c_2, w_1(A), w_1(B), r_1(A), r_1(B), r_1(A), c_1$ .....

.....

ii) Is it possible that an *unrepeatable read* happens in T_1?

Unrepeatable Read in T_1: ⊗ yes ○ no

`Possible solution::` .....

$b_1, b_2, r_1(A), w_1(A), w_1(B), r_1(A), w_2(A), c_2, r_1(B), r_1(A), c_1$ .....

.....

b) In addition to T_1 from a), consider the transaction

$$T_3 \colon b_3, w_3(C), r_3(B), w_3(B), r_3(A), c_3$$

running in isolation level L2.

Is there a valid schedule of these two transactions that is not resettable?

Schedule that is not resettable: ○ yes ⊗ no

$b_1, b_3, r_1(A), w_1(A), w_1(B), r_1(A), r_1(B), w_3(C), r_3(B), w_3(B), r_3(A), c_3, r_1(A), c_1$ .....

.....

.....

MatrikelNr: _____ **Last name:** _____

Question 2: Logging and Recovery (12)

Consider the given Log/Compensation records of transactions T_1, T_2, T_3, and T_4.
Perform a recovery following the ARIES algorithm using these records.

Log entries (Archive)

[#1, T_1, BOT,			#0]
[#2, T_4, BOT,			#0]
[#3, T_4, P_D,	D+=3,	D–=3,	#2]
[#4, T_2, BOT,			#0]
[#5, T_2, P_A,	A–=2,	A+=2,	#4]
[#6, T_2, P_B,	B–=4,	B+=4,	#5]
[#7, T_3, BOT,			#0]
[#8, T_4, P_D,	D+=1,	D–=1,	#3]
[#9, T_3, P_C,	C+=9,	C–=9,	#7]
[#10, T_4, P_A,	A+=2,	A–=2,	#8]
[#11, T_4, P_B,	B–=2,	B+=2,	#10]
[#12, T_1, P_B,	B+=4,	B–=4,	#1]
[#13, T_2, P_A,	A+=15,	A–=15,	#6]
⟨#14, T_4, P_B,	B+=2,	#11,	#10⟩
[#15, T_1, COMMIT,			#12]
⟨#16, T_4, P_A,	A–=2,	#14,	#8⟩
[#17, T_3, P_C,	C+=3,	C–=3,	#9]

Pages on the persistent storage:

P_A LSN: #5
$A = 20$

P_B LSN: #6
$B = 23$

P_C LSN: #9
$C = 40$

P_D LSN: #8
$D = 17$

a) Compute the LSNs for P_A, P_B, P_C, P_D and values for A, B, C, D after the redo phase. (4 points)

P_A LSN: #16
$A =$ 35

P_B LSN: #14
$B =$ 27

P_C LSN: #17
$C =$ 43

P_D LSN: #8
$D =$ 17

b) Perform the undo phase. List the compensation log records (CLRs) in the lines below. You might not need all lines. Employ the usual format, which is also stated again on the first page of the exam. (6 points)

⟨#18, T_3, P_C, C-=3, #17, #9⟩ .....

⟨#19, T_2, P_A, A-=15, #13, #6⟩ .....

⟨#20, T_3, P_C, C-=9, #18, #7⟩ .....

⟨#21, T_4, P_D, D-=1, #16, #3⟩ .....

⟨#22, T_3, BOT, #20⟩ .....

⟨#23, T_2, P_B, B+=4, #19, #5⟩ .....

⟨#24, T_2, P_A, A+=2, #23, #4⟩ .....

⟨#25, T_2, BOT, #24⟩ .....

⟨#26, T_4, P_D, D-=3, #21, #2⟩ .....

⟨#27, T_4, BOT, #26⟩ .....

c) State the values for A, B, C, and D after the undo phase has finished. (2 points)

A: 22 ..... B: 31 ..... C: 31 ..... D: 13 .....

Question 3: Locking/Concurrency Control (10)

Below, a sequence of Exclusive- and Share Locks, abbreviated by $X_i(O)$ and $S_i(O)$, releases of locks, abbreviated by $relX_i(O)$ or $relS_i(O)$, read- and write operations is given, abbreviated by $r_i(O)$ and $w_i(O)$, begin of transaction, abbreviated by b_i, end of transaction, abbreviated by c_i or a_i. For different entries within the same row, an arbitrary order may by assumed. A lock upgrade is allowed (shared-lock to exclusive-lock). Then, only the exclusive lock needs to be released.

(3a) 2-Phase Locking (2PL): Correct Protocol?
State whether each of the five transactions T_1, T_2, T_3, T_4, and T_5 follows the *2-Phase Lock (2PL)* protocol. If not, describe how 2PL is violated. (5 points)

T_1	T_2	T_3	T_4	T_5
b_1	b_2	b_3	b_4	b_5
	$S_2(A)$		$S_4(A)$	$X_5(C)$
$S_1(A)$	$r_2(A)$			$w_5(C)$
$r_1(A)$				
				$X_5(B)$
	$r_2(A)$			$relX_5(C)$
	$X_2(C)$		$r_4(A)$	$w_5(B)$
	$relS_2(A)$			
	$w_2(C)$			$relX_5(B)$
$S_1(B)$	$w_2(C)$		$S_4(B)$	
	$relX_2(C)$		$r_4(B)$	
			$X_4(C)$	$X_5(D)$
$r_1(B)$			$w_4(C)$	$w_5(D)$
		$S_3(C)$		$relX_5(D)$
$X_1(D)$		$r_3(C)$		c_5
$w_1(D)$	c_2	$X_3(C)$	$relS_4(B)$	
$relX_1(D)$		$w_3(C)$	$relS_4(A)$	
$relS_1(B)$		$relX_3(C)$	$relX_4(C)$	
c_1		a_3	c_4	

T_1 does not release A .....

T_2: correct 2PL. .....

T_3 locks C exklusiv (XL), ...

while T_4 holds the exklusive lock ..

T_4: correct 2PL. .....

T_5: does not have XL on D but .....

releases a lock. .....

.....

.....

.....

.....

.....

.....

Hint: Consider not only one transaction at a time, but also transactions together.

(3b) Strict 2-Phase Lock (Strict 2PL) Protocol: Determine Locks

In order to enforce multi-user synchronization when accesses to a database occur, the strict 2-phase lock protocol is used. Consider the schedule given below.

Provide a sequence of lock requests and releases. If a transaction waits explicitly state that the transaction waits. If a transaction is in deadlock at the end of the schedule, assume that the transaction is aborted. In that case, release objects for which a lock has been granted.

*You might find the notation as given on the previous page useful (**X**$_i(O)$, **S**$_i(O)$, **relX**$_i(O)$, **relS**$_i(O)$). In addition, use **gS**$_i(O)$ and **gX**$_i(O)$ to denote that read- or write lock is granted to transaction T_i. Take **wait**$_i$ to indicate that transaction T_i waits and use cont$_i$ to state that transaction T_i continues. You may omit continue, but we suggest that you add it for better readability. Use **abort**$_i$ to state that Transaction T_i was aborted.*

(5 points)

T_1	T_2	T_3	T_4
			b_4
		b_3	
b_1			
$r_1(A)$			
		$r_3(A)$	
$w_1(B)$			
	b_2		
	$r_2(A)$		
$w_1(A)$			
			$r_4(C)$
	$w_2(C)$		
			a_4
	$w_2(D)$		
	c_2		
		$r_3(B)$	

$\mathtt{S}_1(A)$,$\mathtt{gS}_1(A)$,$\mathtt{S}_3(A)$,$\mathtt{gS}_3(A)$,$\mathtt{X}_1(B)$,$\mathtt{gS}_1(B)$,$\mathtt{S}_2(A)$,$\mathtt{gS}_2(A)$,

$\mathtt{X}_1(A)$,$\mathtt{wait}_1$,$\mathtt{S}_4(C)$,$\mathtt{gS}_4(C)$,$\mathtt{X}_2(C)$,$\mathtt{wait}_2$,$\mathtt{relS}_4(C)$,$\mathtt{abort}_4$,

$\mathtt{cont}_2$,$\mathtt{gX}_2(C)$,$\mathtt{X}_2(D)$,$\mathtt{gX}_2(D)$,$\mathtt{relX}_2(D)$,$\mathtt{relX}_2(C)$,$\mathtt{relS}_2(A)$,

$\mathtt{S}_3(B)$,$\mathtt{wait}_3$,$\mathtt{relS}_3(B)$,$\mathtt{relS}_3(A)$,$\mathtt{abort}_3$, ...

$\mathtt{X}_1(A)$,$\mathtt{relX}_1(A)$,$\mathtt{relX}_1(B)$,$\mathtt{abort}_1$ .....

.....

.....

.....

.....

.....

.....

Questions 4–6 are all based on the database schema described on this page.

Question 4: Defining a database schema using SQL and dependencies (12)

a) The following schema is given

`players`	(name, pid, country, mostGoals: *contracts.id*)
`contracts`	(id, playerName: *players.name*, playerId: *players.pid*, salary, club)
`games`	(id: *contracts.id*, gid, coach: *coaches.name*, goals, time)
`coaches`	(name, totalGoals, country)

A university assistant needs to find a relational schema for a question in a data base systems test and, in the absence of new ideas, decides upon a generic schema consisting of players, contracts, games and coaches.

A player is uniquely identified by a `name` and a player id `pid`. In addition, the `country` of the player is also stored.

Players have contracts (`contracts`) with different clubs. Each contract is uniquely identified by an `id`. The ID must be a sequence, starting with 5 and continuing in steps of 10. The player who has signed the contract is also saved (`playerName, playerId`). The `salary` of the player and the name of the club (`club`) are also recorded. `salary` should be implemented as a decimal number with at most 2 decimal places.

In addition, the contract under which a player has scored the most goals (`mostGoals`) is recorded for each player. *For the solution of this task it is not necessary to check if `mostGoals` really refers to the contract under which the most goals were scored.*

A player plays in games (`games`) for each contract. Every game is uniquely identified by the combination of the `id` of the associated contract and a game id (`gid`). For each game the number of achieved goals by the corresponding player of the contract is stored (`goals`), how many minutes they player was playing (`time`), as well as the `coach`. `goals` must be at least 0.

Each coach has a unique `name`. In addition, the total number of scored goals of associated players (`totalGoals`) and the `country` of the coach are recorded in the database. The number of the total goals must lie between 0 and 5000 for each entry in the table. *For the solution of this assignment it is not necessary to check if `totalGoals` really is set to the total number of associated goals.*

Provide the necessary SQL statements to create the described schema, with all described integrity constraints. Choose appropriate types for attributes. **You may use VC in place of VARCHAR(100).**

(8 points)

```sql
CREATE SEQUENCE seq_id INCREMENT BY 10 MINVALUE 5 NO CYCLE;

CREATE TABLE players(
   name         VARCHAR(100),
   pid          INTEGER,
   country      VARCHAR(100),
   mostGoals    INTEGER,
   PRIMARY KEY (name,pid)
);

CREATE TABLE contracts(
   id           INTEGER DEFAULT nextval('seq_id') PRIMARY KEY,
   playerName   VARCHAR(100),
   playerId     INTEGER,
   salary       NUMERIC(8,2) NOT NULL,
   club         VARCHAR(100) NOT NULL,
   FOREIGN KEY (playerName,playerId) REFERENCES players(name,pid)
);

ALTER TABLE players ADD CONSTRAINT c_mostGoals
   FOREIGN KEY (mostGoals) REFERENCES contracts(id)
   DEFERRABLE INITIALLY DEFERRED;


CREATE TABLE coaches(
   name         VARCHAR(100) PRIMARY KEY,
   totalGoals   INTEGER NOT NULL CHECK(totalGoals BETWEEN 0 AND 5000),
   country      VARCHAR(100) NOT NULL
);

CREATE TABLE games(
   id           INTEGER REFERENCES contracts(id),
   gid          INTEGER,
   coach        VARCHAR(100) REFERENCES coaches(name),
   goals        INTEGER NOT NULL CHECK(goals >= 0),
   time         INTEGER NOT NULL,
   PRIMARY KEY (id,gid)
);
```

Hint: Take care of the order of your statements.

b) For this task, you have to consider different key-dependencies.

(4 points)

i) Consider the following schema:

`game` (id: *level.game*, hardestLevel: *level.name*)
`level` (game: *game.id*, name)

Complete the following tables in such a way **that the resulting database instance satisfies the schema.** You must assume that the two tables represent the complete database. To get points, the tables must be filled completely.

level

game	name
220	Blocks
330	Stone Age
100	Space
450	Tunnels
510	Mushrooms

game

id	hardestLevel
510	Mushrooms
220	Blocks
100	Space
330	Stone Age
450	Tunnels

Note: The solution is not unique, you need to enter only one possible solution. **Note:** There are multiple ways of solving this subtask, and only one possible solution is shown here.

ii) Consider the following schema:

`Q` (id, W: *W.id*, E: *E.id*)
`W` (id, Q: *Q.id*)
`E` (id, W: *W.id*)

Complete the following tables in such a way **that the resulting database instance satisfies the schema.** You must assume that the three tables represent the complete database. To get points, the tables must be filled completely.

Q

id	W	E
1	2	1
2	3	3
3	1	2

W

id	Q
1	3
2	2
3	1

E

id	W
1	1
2	2
3	3

Note: The solution is not unique, you need to enter only one possible solution. **Note:** There are multiple ways of solving this subtask, and only one possible solution is shown here.

Question 5: Recursive Queries (12)

You are given the following recursive query using the same database schema as was used in question 4 a) :

```
WITH RECURSIVE tmp(playerName, country) AS
(
 SELECT  p.name, p.country
 FROM    players p
 WHERE   p.name = 'Valentina Sommer' AND p.country = 'United Kingdom'
UNION
 SELECT  p.name, p.country
 FROM    tmp t, contracts c1, contracts c2, players p, games g1, games g2
 WHERE   t.playerName = c1.playerName
         AND c1.id = g1.id
         AND c2.id = g2.id
         AND g1.time = g2.time
         AND c2.playerName = p.name
)
SELECT  t.playerName, t.country
FROM    tmp t
```

Evaluate the given query over the database instance, which can be found on Page 13. We suggest to unstaple Page 13 for easier handling.

playerName	country
Valentina Sommer	United Kingdom
Juliane Dietrich	Austria
Frank Baumann	United States
Jakob Mueller	Netherlands

Question 6: PL/SQL Trigger (12)

Consider the database instance given on **Page 13** (last sheet of the exam). We suggest to unstaple that page.

Each of the sub-tasks contains an SQL statement, which has to be evaluated on the **given instance**. For each task assume that the database was reset and only the content of the last page is relevant, meaning, an insert statement from sub-task (a) has no effect on sub-task (b) etc. Moreover, each function and trigger is only relevant for the corresponding sub-task.

Provide the result of the `SELECT`-Statements. If an error occurs, provide an explanation.

You are free to use arbitrary shorthand notations in your answers (as long as they can be uniquely identified).

a) (4 points)

```
CREATE OR REPLACE FUNCTION fInsertGamesTwo() RETURNS TRIGGER AS $$
BEGIN

        UPDATE coaches SET totalGoals = totalGoals + NEW.goals
        WHERE NEW.coach = name;

        RETURN NEW;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tInsertGamesTwo AFTER INSERT ON games
       FOR EACH ROW EXECUTE PROCEDURE fInsertGamesTwo();
```

```
INSERT INTO games(id, gid, coach, goals, time)
VALUES (65, 4, 'Smith', 5, 30),
       (15, 4, 'Wilson', 1, 93),
       (15, 5, 'Wilson', 1, 95),
       (25, 5, 'Miles', 2, 80);

SELECT * FROM coaches
  WHERE country = 'United Kingdom' AND totalGoals > 0;
```

```
name | totalgoals | country
--------+------------+----------------
Tilden | 2 | United Kingdom
Smith | 6 | United Kingdom
Wilson | 5 | United Kingdom
Miles | 2 | United Kingdom
```

b) (4 points)

```
CREATE OR REPLACE FUNCTION fInsertGames() RETURNS TRIGGER AS $$
DECLARE
        old_g INTEGER;
        curr_pid INTEGER;
        new_g INTEGER;
BEGIN
        SELECT SUM(g.goals), p.pid
               INTO old_g, curr_pid
        FROM games g, players p, contracts c
        WHERE NEW.id = c.id AND c.playerId = p.pid AND
              p.mostGoals = g.id
        GROUP BY p.pid;

        SELECT SUM(goals) INTO new_g
        FROM games WHERE NEW.id = id;

        IF new_g > old_g THEN
                UPDATE players SET mostGoals = NEW.id
                WHERE curr_pid = pid;
        END IF;
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER tInsertGames AFTER INSERT ON games
       FOR EACH ROW EXECUTE PROCEDURE fInsertGames();
```

```
INSERT INTO contracts(id, playerName, playerId, salary, club)
VALUES (70, 'Frank Baumann', 12, 200000.50, 'FC Barca'),
       (75, 'Valentina Sommer', 13, 100000.50, 'Real Madrid'),
       (80, 'Juliane Dietrich', 11, 300000.50, 'FC Barca'),
       (85, 'Thomas Stidl', 14, 400000.50, 'Real Madrid');

INSERT INTO games(id, gid, coach, goals, time)
VALUES (70, 4, 'Martins', 1, 50),
       (75, 4, 'Smith', 2, 70),
       (75, 5, 'Smith', 2, 70),
       (80, 4, 'Wilson', 4, 70),
       (85, 5, 'Taylor', 1, 70);

SELECT name, pid, mostGoals FROM players WHERE pid < 15;
```

```
name | pid | mostgoals
-----------------+-----+-----------
Thomas Stidl | 14 | 45
Frank Baumann | 12 | 5
Valentina Sommer | 13 | 75
Juliane Dietrich | 11 | 80
```

c) (4 points)

```
CREATE OR REPLACE FUNCTION fExpensiveContracts() RETURNS TRIGGER AS
$$
BEGIN

    IF NEW.club = 'Real Madrid' AND NEW.salary > 400000 THEN
        INSERT INTO contracts
        VALUES (NEW.id, NEW.playerName, NEW.playerId,
                NEW.salary / 2, NEW.club);
        RETURN NULL;
    ELSE
        RETURN NEW;
    END IF;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tExpensiveContracts
    BEFORE INSERT
    ON contracts
    FOR EACH ROW
EXECUTE PROCEDURE fExpensiveContracts();
```

```
INSERT INTO contracts(id, playerName, playerId, salary, club) VALUES
  (70, 'Frank Baumann', 12, 500000.00, 'FC Barca'),
  (75, 'Valentina Sommer', 13, 840000.00, 'Real Madrid'),
  (80, 'Juliane Dietrich', 11, 800000.00, 'FC Barca'),
  (85, 'Thomas Stidl', 14, 600000.00, 'Real Madrid');


SELECT id, salary, playerid FROM contracts
  WHERE club = 'FC Barca' OR club = 'Real Madrid';
```

```
 id | salary | playerid
----+-----------+----------
70 | 500000.00 | 12
75 | 210000.00 | 13
80 | 800000.00 | 11
85 | 300000.00 | 14
```

Overall: 70 points

Have a successful exam.

MatrikelNr: _____ **Last name:** _____

Instance for Question 5 and Question 6:

You may separate this page form the exam and keep this page.

Please do not write on this page! Solutions written on this sheet will not be graded!

players

name	pid	country	mostGoals
Juliane Dietrich	11	Austria	25
Valentina Sommer	13	United Kingdom	15
Thomas Stidl	14	Germany	45
Frank Baumann	12	United States	5
Alice Wallner	15	Italy	35
Elisabeth Stoll	16	Belgium	65
Jakob Mueller	17	Netherlands	55

games

id	gid	coach	goals	time
5	1	Smith	1	90
5	2	Smith	0	40
15	1	Wilson	2	55
15	2	Wilson	1	90
15	3	Taylor	0	70
25	1	Miles	0	55
35	1	Hemming	1	45
45	1	Tilden	2	20
55	1	Martins	0	10
55	2	Martins	1	70
65	1	Smith	0	85

contracts

id	playerName	playerId	salary	club
5	Frank Baumann	12	500000.5	FC Lions
15	Valentina Sommer	13	450000.75	SK Castle
25	Juliane Dietrich	11	300000.9	FC Bulls
35	Alice Wallner	15	250000.5	FC Pool
45	Thomas Stidl	14	450000.65	SK Tempo
55	Jakob Mueller	17	400000.56	SK Burg
65	Elisabeth Stoll	16	200000.55	FC Sun

coaches

name	totalGoals	country
Smith	1	United Kingdom
Wilson	3	United Kingdom
Martins	1	Austria
Taylor	0	United Kingdom
Miles	0	United Kingdom
Hemming	1	Germany
Tilden	2	United Kingdom