Gruppe **A**

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

EXAM AUS		10.03.2020
◯ DATA MODELLING 2 (**184.790**)	◯ DATABASE SYSTEMS (**184.686**)	**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!**

Attention! The following rule applies to all multiple choice questions: Just crossing/checking "Yes" or "No" does not give any points; points are only awarded in combination with the specified justification/example/...

Exercise 1: Locking Protocols (14)

Assume some DBMS implements the Isolation Levels using the following locking protocols:

1. **Read Uncommited (RU):** MGL with 2PL for Exclusive-Locks, no locks for read operations.
2. **Read Committed (RC):** MGL with 2PL for Exclusive-Locks, MGL without 2PL for Share-Locks (Share locks are requested before each read operation and released immediately after the read).
3. **Repeatable Read (RR):** MGL with 2PL (no restriction/special rules).
4. **Serializable (Ser):** MGL with 2PL and just a single level: complete Database.

Multiple Granularity Locking for RU, RC, and RR works on the locking levels database, area, page, and field.

As depicted on **"Page Y"** (end of the exam sheet), the database DB is structured in two areas α and β with the pages P_A, respectively P_C and P_E, each with the associated fields.

a) Assuming all transactions on the DBMS run in isolation level **Serializable**, is it possible for a deadlock to occur in such a situation?
If yes, state a schedule which contains a deadlock (use the notation as specified in task b).
If not, provide a short (1-2 sentences) justification.

Deadlock possible if only Serializable: ◯ yes ◯ no

.....

.....

b) Consider the schedule shown below. For each of the four transactions T_1, T_2, T_3 and T_4, the **isolation level** used by the transaction is also stated.

	T_1 **(RR)**	T_2 **(RC)**	T_3 **(Ser)**	T_4 **(RR)**
1	b_1	b_2	b_3	b_4
2		$r_2(B)$		
3	$r_1(F)$			
4				$r_4(B)$
5		$w_2(A)$		
6			$w_3(F)$	
7			$r_3(C)$	
8	$w_1(E)$			
9		$w_2(P_C)$		
10			$w_3(G)$	
11		$w_2(G)$		
12				$r_4(P_E)$
13		$r_2(E)$		
14				$w_4(\alpha)$
15		$w_2(E)$		
16			$r_3(D)$	
17			$w_3(E)$	
18				$r_4(A)$
19	$w_1(B)$			
20				c_4
⋮	⋮	⋮	⋮	⋮

Notation:

- $r_i(O)/w_i(O)$: Read-/write operation of T_i on object O.
- b_i/ c_i: Start/commit of T_i.

i) Which line must *at least* have been processed before T_4 is allowed to release the lock on P_E?

Release after line .....

ii) Assume that after step 20 transaction T_2 releases the lock on G. Which of the fields A–G can T_2 read resp. write before it commits?

Read fields: .....

Write fields: .....

iii) State the wait-for graph ("Wartegraphen") for the situation immediately after line 14.

iv) For the following objects, state all locks held by each transaction immediately after line 14.
Use S, X, IS and IX if a transaction holds a corresponding lock, and WS, WX, WIS and WIX if the transaction needs to wait for the corresponding lock.

	T_1	T_2	T_3	T_4
E:				
P_E:				

	T_1	T_2	T_3	T_4
α:				
DB:				

Exercise 2: Logging and Recovery (9)

Conduct a recovery following the ARIES procedure based on the log records and content of the pages shown below. The log format is the one used in the lecture and exercises, a summary of the format is provided on **"Page Y"** at the end of the exam.

a) List all the *log records* created during the recovery.

Log records (given)

[#1, T_2, `BOT`, #0]
[#2, T_3, `BOT`, #0]
[#3, T_3, P_C, C+=15, C-=15, #2]
[#4, T_1, `BOT`, #0]
[#5, T_2, P_A, B+=1, B-=1, #1]
[#6, T_4, `BOT`, #0]
[#7, T_2, P_E, E+=15, E-=15, #5]
[#8, T_4, P_A, B+=5, B-=5, #6]
[#9, T_1, P_E, E-=16, E+=16, #4]
[#10, T_3, P_A, A+=20, A-=20, #3]
[#11, T_3, P_A, A+=1, A-=1, #10]
⟨#12, T_2, P_E, E-=15, #7, #5⟩
[#13, T_4, P_C, C-=2, C+=2, #8]
[#14, T_4, P_A, A-=5, A+=5, #13]
⟨#15, T_2, P_A, B-=1, #12, #1⟩
[#16, T_1, P_E, E-=10, E+=10, #9]
⟨#17, T_4, P_A, A+=5, #14, #13⟩
⟨#18, T_3, P_A, A-=1, #11, #10⟩
⟨#19, T_2, `BOT`, #15⟩

Log records (your answer)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Pages (content of DB)

P_A	LSN: #10
$A = 12$ $B = 15$	

P_C	LSN: #8
$C = 20$ $D = 25$	

P_E	LSN: #7
$E = 10$ $F = 15$ $G = 11$	

b) State the values of the fields A, B, C, and E *after* the *Redo*-Phase:

A: ..... B: ..... C: ..... E: .....

c) State the values of the fields A, B, C, and E *after* the successful *recovery*:

A: ..... B: ..... C: ..... E: .....

Exercise 3: Properties of Transactions (12)

Consider the scenario from Task 1 (hypothetical DBMS and structure of the database; not the schedule), and assume that all transactions run using the isolation level **Read Committed** only.

Answer the following questions by checking "Yes" or "No". *If you answer "Yes"*, briefly justify your answer (1–2 sentences). *If you answer "No"* state a schedule (operations: b_i, c_i, a_i, $r_i(O)$, $w_i(O)$), that is consistent with the **"implementation"** of the isolation level, but violates the property under consideration.
Note: You need not state lock requests or releases of locks – just make sure that your schedule is consistent with some valid (according to the protocol) sequence of locks and releases. If such a sequence exists, we will find it :)

a) Assume there are only write operations. Are all schedules produced by the DBMS *conflict serializable*?

Only writes → conflict serializable: ◯ yes ◯ no

.....

.....

.....

b) Assume there are read- and write operations. Are all schedules produced by the DBMS *conflict serializable*?

Reads and Writes → conflict serializable: ◯ yes ◯ no

.....

.....

.....

c) Assume there are only write operations. Are all schedules produced by the DBMS *strict*?

Only writes → strict: ◯ yes ◯ no

.....

.....

.....

d) Assume there are read- and write operations. Are all schedules produced by the DBMS *recoverable*?

Reads and Writes → Recoverable: ◯ yes ◯ no

.....

.....

.....

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

Exercise 4: Defining a database schema using SQL (7)

Consider the following schema:

importer(id, country)
producer(brand, origin, name, family: *(fruit.name, fruit.family)*)
fruit(name, family, mainProducer: *(producer.brand)*)
purchases(id: *importer.id*, producer: *producer.brand*, name, family: *(fruit.name, fruit.family)*)

Every importer has a unique id, in addition to this their country of origin is also stored. Implement a sequential enumeration for the ID with the help of a Sequence. This Sequence should start at 100, and increase in steps of 10. For a producer, the brand and origin is stored, where the brand must be unique. Furthermore, the fruit a producer exported the most of is also stored explicitly. Each fruit is identifiable via the pair of name and family, additionally the main exporter of the fruit is also stored. The purchases relation states which importer purchases which fruit from which producer.

Provide the necessary SQL commands to create database tables according to the provided schema. Make sure to implement all of the described integrity constraints. You may choose appropriate attributes for the columns.
You may abbreviate VARCHAR(100) with VC.

Hint: Take care of the order of your statements.

Exercise 5: Recursive Queries (14)

Consider the recursive SQL query over the database schema given in "Exercise 4".

```
WITH RECURSIVE tmp(name,family) AS
(
SELECT  name,family
FROM    producer
WHERE   brand = 'SanLucar'
UNION ALL
SELECT  k.name,k.family
FROM    producer p, purchases k, fruit f NATURAL JOIN tmp t
WHERE   p.brand = f.mainProducer AND k.producer = p.brand AND (k.id / 10) % 2 = 0
)
SELECT name FROM tmp GROUP BY name,family;
```

Evaluate this query over the database instance given on the last page of the exam:

Exercise 6: PL/SQL Trigger (14)

Whenever a tuple K is deleted from `purchases`, the `fruit` relation shall be adapted accordingly. Write a PL/pgSQL trigger `trF` and the associated procedure to implement the following behavior:

- Find the fruit F referenced by the corresponding foreign key in K.
- If F.`MainProducer` is equal to K.`Producer`, the value of `MainProducer` in F shall be updated as follows:
  - If the value of `family` in F is "Nuss", one of the producers selling F shall be selected randomly as new `MainProducer` of F. (A producer sells a fruit if both occur within the same tuple in `purchases`.)
  - Otherwise the new `MainProducer` of F shall be the producer with the alphabetically smallest value in field `origin` among all producers selling F.
  - You may assume that a suitable producer always exists.
- Make sure, that the deleted entry K is not taken into account when choosing the new value for `MainProducer`.

Overall: 70 points

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

Thus, please do not provide any solutions on this page! Solutions written on this sheet will not be graded!

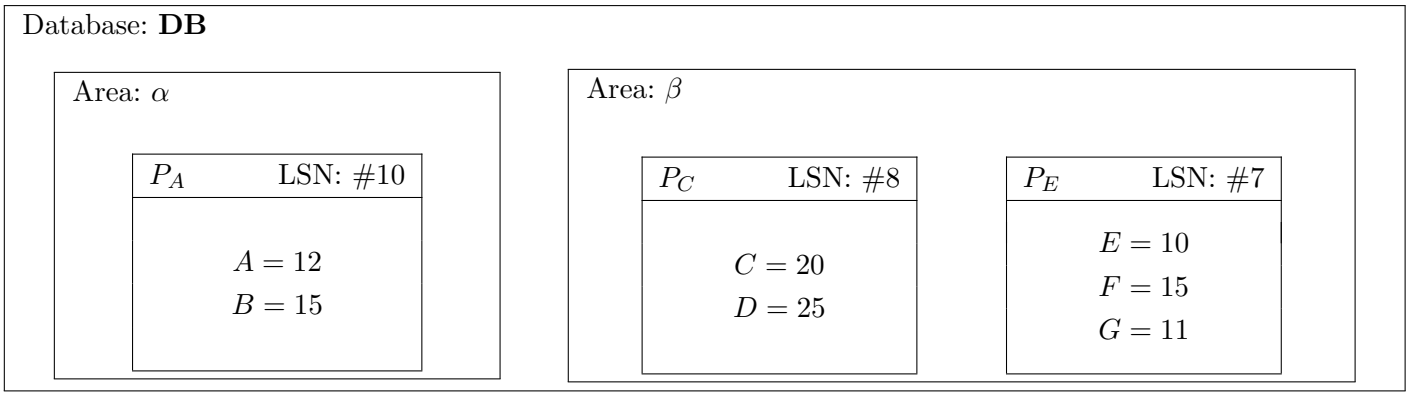
(Additional Information/Content for Tasks 1–3)

Description of the format of the log entries (Task 2):

We write `[LSN, TA, PageID, Redo, Undo, PrevLSN]` *for common log records, and* ⟨`LSN, TA, PageID, Redo, PrevLSN, UndoNextLSN`⟩ *for compensation log records. For* `BOT` *and* `COMMIT` *records, the shortened forms* `[LSN, TA, BOT, PrevLSN]` *resp.* `[LSN, TA, COMMIT, PrevLSN]` *may be used, and accordingly for CLRs, e.g.* ⟨*LSN, TA, BOT, PrevLSN*⟩ *for BOT-CLRs.*

Please note that the Undo/Redo records are given ***relatively*** *to the current value in the database using* ***summation*** *and* ***subtraction****. For example, [#i, T_j, P_X, X+=d_1, X-=d_2, #k] describes that according to the log record with LSN #i the transaction T_j wrote the field X, which is located on page P_X, and that for a redo the value of X needs to be increased by d_1 and for an undo the value of X needs to be reduced by d_2. Finally, the previous log record of this transaction has the LSN k.*

Structure and Content of the Database (Task 1 – 3):



Sample instance for Task 5:

importer

id	country
100	Deutschland
120	Niederlande
130	Frankreich
140	Österreich
160	Belgien

producer

brand	origin	name	family
SanLucar	Spanien	Dwarf Cavendish	Banane
TerraSol	Ecuador	Pernambumco	Ananas
Kailas	Indien	Cashew	Nuss
Calavo	USA	Pinkerton	Avocado
EKM	Südafrika	Sanguinello	Orange

fruit

name	family	mainProducer
Dwarf Cavendish	Banane	SanLucar
Blue Java	Banane	TerraSol
Red Dacca	Banane	Kailas
McIntosh	Apfel	Calavo
Akane	Apfel	SanLucar
Pernambumco	Ananas	TerraSol
Sanguinello	Orange	SanLucar
McIntosh	Orange	TerraSol
Pinkerton	Avocado	Calavo
Cashew	Nuss	Kailas

purchases

id	producer	name	family
100	SanLucar	Pinkerton	Avocado
100	SanLucar	McIntosh	Orange
120	SanLucar	Blue Java	Banane
140	SanLucar	McIntosh	Apfel
160	SanLucar	Cashew	Nuss
130	Calavo	Sanguinello	Orange
140	TerraSol	Cashew	Nuss
140	EKM	Red Dacca	Banane
100	EKM	Pernambumco	Ananas

Good luck and a successful start into the new term!