Gruppe **A**

Bitte tragen Sie **SOFORT** und **LESERLICH** Namen und Matrikelnr. ein, und legen Sie Ihren Ausweis bereit.

PRÜFUNG AUS			21.01.2021
◯ DATENMODELLIERUNG 2 (**184.790**)		◯ DATENBANKSYSTEME (**184.686**)	**GRUPPE A**
Matrikelnr.	Familienname	Vorname	

Arbeitszeit: 90 Minuten. Lösen Sie die Aufgaben auf den vorgesehenen Blättern; Lösungen auf Zusatzblättern werden nicht gewertet. **Viel Erfolg!**

Achtung!

Für sämtliche Fragen mit Ankreuzmöglichkeiten gilt: Ankreuzen alleine gibt keine Punkte!

Punkte gibt es nur in Zusammenhang mit geforderter Erklärung/Beispiel/...!

Notation:

In den Aufgaben 1 – 3 wird die folgende (aus der Vorlesung bekannte) Notation für Transaktionen T_i verwendet:

- $r_i(O)$ und $w_i(O)$: Lese- bzw. Schreibzugriff von T_i auf Objekt O.
- b_i, c_i, a_i: Beginn (`BEGIN OF TRANSACTION`), Commit (`COMMIT`) bzw. Abbruch (`ABORT/ROLLBACK`) von T_i.

Die Indizes $_i$ können weggelassen werden, wenn klar ist zu welcher Transaktion eine Operation gehört.

Des weiteren wird das aus der Vorlesung bekannte Format für Logeinträge verwendet:

`[LSN, TA, PageID, Redo, Undo, PrevLSN]` für "normale" Einträge, bzw.
`[LSN, TA, BOT, PrevLSN]` für `BOT` Log-Einträge und
`[LSN, TA, COMMIT, PrevLSN]` für `COMMIT` Einträge.

Kompensations Logeinträge (Compensation Log Records) haben das Format
⟨`LSN, TA, PageID, Redo, PrevLSN, UndoNextLSN`⟩ bzw.
⟨`LSN, TA, BOT, PrevLSN`⟩

Dabei stellt `LSN` die Log-Sequence Nummer dar, `TA` die Transaktion, `PageID` ist die veränderte Seite, `Redo` und `Undo` die für das Redo bzw. Undo benötigten Informationen, `UndoNextLSN` ist die LSN des nächsten Logeintrags der selben Transaktion welcher zurückgesetzt werden soll, und `PrevLSN` die LSN des vorherigen Logeintrags derselben Transaktion.

Im Falle logischer Protokollierung sollen die Änderungen zum aktuellen Datenbestand nur mittels *Addition* bzw. *Subtraktion* angegeben werden, z.B. [·, ·, ·, X+=d_1, X-=d_2, ·].

Aufgabe 1: Eigenschaften von Transaktionen (12)

Betrachten Sie die unten angegebene Historie, welche durch eine Abfolge von Elementaroperationen der vier Transaktionen T_1, T_2, T_3 und T_4 auf den Datensätzen A, B, C und D gegeben ist. *Dabei bezeichnet b_i den Beginn, c_i den erfolgreichen Abschluss (commit), und a_i den Abbruch (abort) von Transaktion T_i. $r_i(O)$ und $w_i(O)$ bezeichnen Lese- bzw. Schreibzugriffe von Transaktion T_i auf das Objekt O.*

T_1	T_2	T_3	T_4
b_1	b_2	b_3	b_4
			$r_4(A)$
		$r_3(A)$	
	$r_2(C)$		
$r_1(D)$			
			$w_4(A)$
		$r_3(B)$	
	$w_2(B)$		
		$w_3(C)$	
$w_1(B)$			
			$w_4(D)$
			c_4
	$r_2(A)$		
	$w_2(C)$		
$r_1(A)$			
		$r_3(D)$	
	c_2		
c_1			
		$w_3(C)$	
		$r_3(B)$	
		$w_3(B)$	
		c_3	

a) Geben Sie an, zwischen welchen Transaktionen eine Leseabhängigkeit besteht. D.h., geben Sie für jede Transaktion an, von welchen anderen Transaktionen diese Transaktion liest (falls eine Transaktion von keiner anderen Transaktion liest, streichen Sie das entsprechende Feld bitte durch).

(4 Punkte)

a) Leseabhängigkeiten:

T_1 liest von ..... T_2 liest von .....

T_3 liest von ..... T_4 liest von .....

b) Bestimmen Sie anschließend, ob die Historie kaskadierendes Rücksetzen vermeidet oder nicht, sowie ob sie strikt ist oder nicht. Geben Sie jeweils eine kurze Begründung an Hand der Historie, an.
(*Achtung:* Ankreuzen alleine ohne eine Begründung gibt keine Punkte!)

(4 Punkte)

b) Eigenschaften:

Historie vermeidet kaskadierendes Rücksetzen: ◯ ja ◯ nein

Begründung: .....

.....

Historie ist strikt: ◯ ja ◯ nein

Begründung: .....

.....

c) Betrachten Sie jeweils die unten angegebenen Paare von Transaktionen. Bestimmen Sie für jedes der beiden Paare, ob die beiden Transaktionen konfliktserialisierbar sind oder nicht.
Falls nicht, geben Sie eine Sequenz von Paaren von Konfliktoperationen der beiden Transaktionen an, welche die Existenz einer konfliktäquivalenten, seriellen Historie ausschließen.
Falls die Transaktionen konfliktserialisierbar sind, geben Sie eine konfliktäquivalente, serielle Historie (= Sequenz von Elementaroperationen!) der beiden Transaktionen an.

(4 Punkte)

Transaktionen T_2 und T_3: Konfliktserialisierbar: ◯ ja ◯ nein

Antwort: .....

.....

Transaktionen T_2 und T_4: Konfliktserialisierbar: ◯ ja ◯ nein

Antwort: .....

.....

Aufgabe 2: Protokollierung und Wiederanlauf (Logging und Recovery) (12)

Betrachten Sie die unten angegebenen Logeinträge der vier Transaktionen T_1, T_2, T_3 und T_4.

a) Führen Sie anhand dieser Log-Einträge ein Recovery (nach dem ARIES Verfahren) durch, und geben Sie die dabei entstehenden Log-Einträge an. (6 Punkte)

Logeinträge (Angabe)

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

Logeinträge (Lösung)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

b) Ist es möglich, anhand der gegebenen Logeinträge zu bestimmen, ob die zugehörige Historie strikt (im Sinne der Klassifikation von Historien im Rahmen der Mehrbenutzersynchronisation) ist? (4 Punkte)
Falls ja, geben Sie an ob die zugehörige Historie strikt war oder nicht (und warum).
Falls nein, begründen Sie kurz (1-2 Sätze) Ihre Antwort.

Strikt kann erkannt werden: ◯ ja ◯ nein

.....

.....

.....

c) Nehmen Sie an, als Ersetzungsstrategie würde *force* und *steal* verwendet.

Bestimmen Sie für die beiden Seiten P_B und P_C möglichst genau die möglichen Werte der LSN (größte LSN deren Änderung auf der Seite durchgeführt wurde) welche am Beginn des Wiederanlaufs vermerkt war (d.h. jenen Wert, der auf der Seite im Hintergrundspeicher vermerkt war, mit welcher das Recovery durchgeführt wurde). (2 Punkte)

Seite	**untere Schranke**		**obere Schranke**
P_A	.....	$\leq$ LSN $\leq$	.....
P_C	.....	$\leq$ LSN $\leq$	.....

Nehmen Sie an dass die LSN beider Seiten ursprünglich `#0` war.

Aufgabe 3: Sperrprotokolle (Locking) (11)

a) Gegeben ist die untenstehende Folge von Exclusive- und Share-Sperren (XL(O), SL(O)), Freigaben von Sperren (relXL(O), relSL(O)), sowie Lese- und Schreiboperationen. Bei Einträgen in der selben Zeile spielt die zeitliche Anordnung der betroffenen Operationen keine Rolle, es kann eine beliebe Reihenfolge angenommen werden.

Geben Sie für jede der fünf Transaktionen T_1, T_2, T_3, T_4 und T_5 an, ob sie das *2-Phasen Sperrprotokoll (2PL)* befolgt. Falls nicht, beschreiben Sie wodurch das 2PL verletzt wird. (5 Punkte)

T_1	T_2	T_3	T_4	T_5
{BOT}	{BOT}	{BOT}	{BOT}	{BOT}
SL(B)	SL(E)			
	SL(B)		XL(D)	
	$r(B)$			SL(B)
				SL(A)
	XL(C)		SL(E)	$r(B)$
	$w(E)$		$r(D)$	relSL(B)
$r(B)$	$r(C)$		$r(E)$	
	relSL(E)			$r(A)$
		XL(A)	relSL(E)	
	$w(C)$	$w(A)$		
	relSL(B)		relXL(D)	XL(E)
	relXL(C)			$w(E)$
		SL(C)		
XL(D)	c_1	$r(C)$		relSL(A)
$w(D)$				
relSL(B)		relXL(A)	c_5	
relXL(D)		relSL(C)		relXL(E)
c_2		c_4		c_3

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Hinweis: Betrachten Sie nicht nur jede Transaktion für sich.

b) Betrachten Sie zwei belibige Transaktionen T_1 und T_2 und einen Datensatz A mit dem initialen Wert w. Weiters nehmen Sie an, dass T_1 den Datensatz A mit dem neuen Wert w' (mit $w' \neq w$) beschreiben möchte und dass T_2 zu einem beliebigen Zeitpunkt vom Datensatz A lesen möchte. Der Wert von A wird ansonsten nicht verändert.

Angenommen beide Transaktionen befolgen das strikte 2-Phasen Sperrprotokoll. Gibt es dann ein Szenario in dem T_2 den neuen Wert w' von A liest und erfolgreich abschließt, wenn T_1 zu einem beliebigen Zeitpunkt abbricht?

Falls ja, geben Sie bitte eine entsprechende Folge von Zugriffen an.
Falls nein, begründen Sie kurz, warum dies nicht passieren kann. (6 Punkte)

○ ja ○ nein

.....

.....

.....

.....

.....

.....

.....

Bitte beachten Sie, dass es nicht ausreicht lediglich eine der Antwortmöglichkeiten anzukreuzen, um auf diese Frage Punkte zu erhalten. Es werden nur Punkte für den korrekten Inhalt Ihrer Antwort verliehen.

Für die Aufgaben 4 – 6 gilt die Datenbankbeschreibung auf diesem Blatt.

Aufgabe 4: Erstellen eines Datenbankschemas mittels SQL und Aufgaben zu Schlüsselabhängigkeiten (11)

a) Folgendes Schema sei gegeben:

`customer`	(name, taxID, country, mostExpensive: *orders.id*)
`orders`	(id, date, fromName: *customer.name*, fromTaxID: *customer.taxID*)
`part`	(id: *orders.id*, number, cost, from: *supplier.name*)
`supplier`	(name, revenue, country)

Ein Universitätsassistent muss für eine Aufgabe in einer Datenbanksysteme Prüfung ein relationales Schema finden, und entscheidet sich mangels neuer Ideen für ein generisches Schema aus Kunden, Lieferanten, Bestellungen und Einzelteilen.

Ein Kunde (`customer`) ist eindeutig gekennzeichnet durch Name und Steuer-ID (`taxID`). Zusätzlich wird das Land (`country`) gespeichert aus dem der Kunde stammt.

Jede der Bestellungen (`orders`) ist eindeutig durch eine ID gekennzeichnet. Die ID muss eine Sequence sein, beginnend mit 100, und in 300er Schritten fortlaufen. Daneben wird auch das Datum (`date`) festgehalten, in dem die Bestellung in Auftrag gegeben wurde. Es wird ebenfalls gespeichert von welchem Kunden die Bestellung stammt (`fromName, fromTaxID`). Daneben wird beim jeweiligen Kunden auch die bisher teuerste Bestellung festgehalten (`mostExpensive`).

Zu jeder Bestellung gibt es eine Reihe an Einzelteilen (`part`). Jedes Einzelteil ist eindeutig gekennzeichnet durch die Kombination der ID der zugehörigen Bestellung und einer Nummer (`number`). Jedes Einzelteil hat auch einen Kostenpunkt (`cost`) und es wird der Lieferant des Einzelteils festgehalten (`from`). Der Kostenpunkt darf nicht negativ sein. `cost` soll als eine Gleitkommazahl mit maximal 2 Kommastellen implementiert werden.

Jeder Lieferant (`supplier`) hat einen eindeutigen Namen. Daneben wird der Umsatz (`revenue`) und das Land (`country`) des Lieferanten in der Datenbank festgehalten. Der Umsatz muss bei jedem Eintrag in der Tabelle zwischen 1.000 und 1.000.000 liegen.

Geben Sie die nötigen SQL Statements an, um obiges Schema (mit allen Konsistenzbedingungen) anzulegen. Wählen Sie passende Typen für Attribute. **Die Abkürzung VC statt VARCHAR(100) ist erlaubt.**

(7 Punkte)

Hinweis: Achten Sie bei den Statements auf die Reihenfolge.

b) Für diese Aufgabe müssen Sie sich mit verschiedenen Schlüsselabhängigkeiten auseinandersetzen.

(4 Punkte)

i) Gegeben ist folgendes Schema:

`shelf` (id: *book.id*, newestBook: *book.name*)
`book` (shelf: *shelf.id*, name)

Es ist gefragt, dass Sie folgende Tabellen so erweitern, **dass die resultierende Datenbank-Instanz das Schema erfüllt**. Hierbei stellen die 2 Tabellen den vollständigen Inhalt der Datenbank dar. Um Punkte für diesen Teil zu erhalten, müssen Sie die Tabellen vollständig ausfüllen.

shelf

id	**newestBook**
3	
	A Dance With Dragons
	The Left Hand of Darkness
5	
	The Colour Of Magic

book

shelf	**name**
2	
	1Q84
4	
	Murder On The Orient Express
1	

ii) Gegeben ist folgendes Schema:

`A` (id, B: *B.id* , C: *C.id*)
`B` (id, C: *C.id*)
`C` (id, B: *C.id*)

Es ist gefragt, dass Sie folgende Tabellen so erweitern, **dass die resultierende Datenbank-Instanz das Schema erfüllt**. Hierbei stellen die 2 Tabellen den vollständigen Inhalt der Datenbank dar. Um Punkte für diesen Teil zu erhalten, müssen Sie die Tabellen vollständig ausfüllen.

A

id	**B**	**C**
1	2	
2		
3	1	2

B

id	**C**
	1
	2
3	

C

id	**B**
1	1
	3
3	

Aufgabe 5: Rekursive Abfragen (12)

Gegeben ist die folgende Rekursive Abfrage auf dem Datenbank-Schema von Aufgabe 4 a):

```
WITH RECURSIVE tmp(id) AS
(
 SELECT  o.id
 FROM    orders o
 WHERE   o.fromName = 'Max Mustermann' AND o.fromTaxID = '12'
UNION
 SELECT  o.id
 FROM    tmp t, orders o, part p, supplier s, customer c
 WHERE   t.id = p.orderID AND p.fromNAME = s.name AND s.country = c.country
         AND c.mostExpensive = o.id AND p.cost < 500 AND s.revenue > 50000
)
SELECT  t.id, c.name
FROM    tmp t, customer c, orders o
WHERE   t.id = o.id AND (o.fromName,o.fromTaxID) = (c.name,c.taxID);
```

Werten Sie diese Abfrage auf der Datenbank-Instanz, die auf der Seite B angegeben ist, aus:

Aufgabe 6: PL/SQL Trigger (12)

Nehmen Sie an, dass die **Funktionen und Trigger** wie auf **Seite T** (am vorletzten Blatt dieser Prüfung) definiert wurden. Die Aufgaben beziehen sich auf die Beispielinstanz auf **Seite B** (letztes Blatt der Prüfung).

In jeder der folgenden Aufgaben ist ein SQL Statement gegeben das **über die Beispielinstanz** ausgeführt wird. (Das Statement in Aufgabe a hat keinen Einfluss auf Aufgabe b usw.) Geben Sie die Ausgabe der `SELECT`-Statements an. **Falls ein Fehler auftreten würde, geben Sie an welcher Fehler auftritt.**

a)

(4 Punkte)

```
INSERT INTO part VALUES (1, 3, 100, 'Rolex');
INSERT INTO part VALUES (1, 4, 200, 'LG');

SELECT name, revenue FROM supplier;
```

---

b)

(4 Punkte)

```
INSERT INTO part VALUES (8, 0, 800, 'LG'), (9, 0, 200, 'Lenovo');

SELECT name, mostexpensive FROM customer;
```

c)

(4 Punkte)

```
INSERT INTO orders VALUES (8, '2021-01-03', 'Otto Normalverbraucher', 14);
INSERT INTO part VALUES (8, 2, 400, 'Lenovo');

SELECT orderid, number, cost FROM part WHERE fromname = 'Lenovo';
```

Gesamtpunkte: 70

Viel Erfolg und ein möglichst reibungsloses und erfolgreiches WS 2021!

Sie können diesen Zettel abtrennen und brauchen ihn nicht abgeben!

Diesen Zettel daher bitte nicht beschriften! (Lösungen auf diesem Zettel werden nicht gewertet!)

Trigger für Aufgabe 6:

```
CREATE FUNCTION setMostExpensive() RETURNS TRIGGER AS $$
DECLARE customerName VARCHAR;
DECLARE customerTaxID INTEGER;
DECLARE lastMostExpensive INTEGER;
DECLARE lastMostExpensiveID INTEGER;
DECLARE currentMostExpensive INTEGER;
DECLARE mostExpensiveOrderID INTEGER;
BEGIN
    SELECT name, taxID, mostExpensive
    INTO customerName, customerTaxID, lastMostExpensiveID
    FROM customer, orders
    WHERE customer.name = orders.fromName AND customer.taxID = orders.fromTaxID
    AND orders.id = NEW.orderID;

    SELECT SUM(cost) INTO lastMostExpensive FROM part
    WHERE part.orderID = lastMostExpensiveID;

    SELECT totalCost, orderID INTO currentMostExpensive, mostExpensiveOrderID FROM (
        SELECT SUM(cost) AS totalCost, id AS orderID FROM part, orders
        WHERE orders.fromName = customerName AND orders.fromTaxID = customerTaxID
        AND orders.id = part.orderID
        GROUP BY id
      ) sq
      ORDER BY totalCost DESC
      LIMIT 1;

    IF currentMostExpensive > lastMostExpensive THEN
        UPDATE customer SET mostExpensive = mostExpensiveOrderID
        WHERE customer.name = customerName AND customer.taxID = customerTaxID;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER trA AFTER INSERT ON part
       FOR EACH ROW EXECUTE PROCEDURE setMostExpensive();
```

```
CREATE FUNCTION increaseRevenue() RETURNS TRIGGER AS $$
DECLARE rev NUMERIC;
```

```
DECLARE rev_sum NUMERIC;
BEGIN

    SELECT revenue INTO rev FROM supplier WHERE name = NEW.fromName;
    SELECT rev + NEW.cost INTO rev_sum;
    IF rev_sum > 1000000 THEN
        RAISE NOTICE 'revenue␣too␣large';
    ELSE
        UPDATE supplier SET revenue = rev_sum WHERE supplier.name = NEW.fromNAME;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER trB AFTER INSERT ON part
       FOR EACH ROW EXECUTE PROCEDURE increaseRevenue();

CREATE FUNCTION splitPart2() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.fromNAME = 'Lenovo' AND NEW.number = 2 THEN
        INSERT INTO part VALUES (NEW.orderID, 3, NEW.cost / 2, NEW.fromNAME), (NEW.orde
        RETURN NULL;
    ELSE
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trC BEFORE INSERT ON part
       FOR EACH ROW EXECUTE PROCEDURE splitPart2();
```

Seite B

Beispielinstanz für Aufgabe 5 und Aufgabe 6:

Sie können diesen Zettel abtrennen und brauchen ihn nicht abgeben!

Diesen Zettel daher bitte nicht beschriften! (Lösungen auf diesem Zettel werden nicht gewertet!)

customer

name	taxID	country	mostExpensive
Max Mustermann	12	Austria	1
Erika Mustermann	13	United States	2
Leon Mustermann	17	Germany	6
Anne Mustermann	18	Germany	7
Otto Normalverbraucher	14	Canada	3
Markus Moeglich	15	Denmark	4
Lieschen Mueller	16	Monaco	5

part

orderID	number	cost	fromName
1	0	100	InGen
1	1	200	Acme Corp.
1	2	300	Yoyodyne
2	0	600	Rolex
3	0	300	Alchemax
5	0	300	LG
5	1	400	Lenovo

orders

id	date	fromName	fromTaxID
1	2021-11-24	Max Mustermann	12
2	2021-10-12	Erika Mustermann	13
3	2020-09-14	Otto Normalverbraucher	14
4	2021-12-01	Markus Moeglich	15
5	2021-11-15	Lieschen Mueller	16
6	2021-11-27	Leon Mustermann	17
7	2020-12-12	Anne Mustermann	18

supplier

name	revenue	country
InGen	100000	United States
Acme Corp.	300000	Canada
Yoyodyne	200000	Denmark
Alchemax	400000	Monaco
Rolex	600000	Switzerland
Lenovo	35000	China
LG	45000	South Korea