

Gruppe A

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

PRÜFUNG AUS		19.06.2020
<input type="radio"/> DATENMODELLIERUNG 2 (184.790)		<input type="radio"/> 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 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 .

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.

In solch einem Eintrag 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, und PrevLSN die LSN des vorherigen Logeintrags derselben Transaktion.

Aufgabe 1: Eigenschaften von Transaktionen

(11)

Gegeben ist die folgende, rücksetzbare Historie der sechs Transaktionen T_1, \dots, T_6 .

	T_1	T_2	T_3	T_4	T_5	T_6	
1	b_1	b_2	b_3	b_4	b_5	b_6	1
2	$r_1(A)$ <input type="text"/>						2
3	$r_1(B)$ <input type="text"/>						3
4			$r_3(A)$ <input type="text"/>				4
5	$w_1(A)$						5
6				$r_4(A)$ <input type="text"/>			6
7		$w_2(C)$					7
8						$r_6(B)$ <input type="text"/>	8
9						$w_6(B)$	9
10				$w_4(A)$			10
11		$r_2(B)$ <input type="text"/>					11
12		$w_2(A)$					12
13						$w_6(C)$	13
14				$r_4(C)$ <input type="text"/>			14
15			$r_3(A)$ <input type="text"/>				15
16				$w_4(C)$			16
17					$r_5(C)$ <input type="text"/>		17
18			$r_3(B)$ <input type="text"/>				18
19	a_1						19

a) Tragen Sie bei den Leseoperationen die Zeilennummer ein, in welcher der gelesene Wert geschrieben wurde. Falls die entsprechende Schreiboperation nicht Teil der Historie ist, tragen Sie bitte ein X ein.

b) Geben Sie für jede Transaktion an, von welchen anderen Transaktionen sie liest.

T_1 liest von
T_2 liest von
T_3 liest von
T_4 liest von
T_5 liest von
T_6 liest von

c) Führen Sie die Historie so fort, dass alle Transaktionen abgeschlossen werden (jede Transaktion T_i kann entweder durch a_i oder c_i abschließen). Die resultierende Historie muss weiterhin rücksetzbar bleiben, und es sollen so viele Transaktionen wie möglich mittels c_i abschließen. Geben Sie dazu für die nächsten 5 Schritte die jeweilige Operation (c_i oder a_i für $2 \leq i \leq 6$) an, und begründen Sie ganz, ganz kurz Ihre Wahl (muss kein vollständiger Satz sein).

Schritt	Operation	Begründung
20
21
22
23
24

Geben Sie eine gültige Sequenz von Logeinträgen an, welche folgende Eigenschaften erfüllt:

- Die Logeinträge sollen von zwei Transaktionen T_1 und T_2 stammen.
- Von beiden Transaktionen sollen die **BEGIN OF TRANSACTION** Einträge enthalten sein.
- T_1 soll mittels **COMMIT** abgeschlossen worden sein, T_2 soll noch aktiv sein.
- Undo und Redo Informationen sollen mittels *physischer* Protokollierung angegeben sein.
- Die Einträge sollen so gewählt sein, dass ein Wiederanlauf (Recovery) mittels dem ARIES Verfahren **nicht** das gleiche Ergebnis liefert, als wenn T_2 nie stattgefunden hätte.
- Sie können die Felder A, B, C, \dots auf den Seiten P_A, P_B, P_C, \dots verwenden. Gehen Sie davon aus, dass alle Felder ursprünglich den Wert 0 besitzen.

.....
.....
.....
.....

b) Unabhängig von a) ist folgender Inhalt der (persistenten) Log-Datei sowie des (flüchtigen) Log-Puffers gegeben.

Log-Datei

Log-Puffer

- | | |
|--------------------------------|---------------------------------|
| [#1, T_1 , BOT, #0] | [#7, T_3 , P_E , ·, ·, #3] |
| [#2, T_2 , BOT, #0] | [#8, T_3 , P_A , ·, ·, #7] |
| [#3, T_3 , BOT, #0] | [#9, T_1 , P_C , ·, ·, #6] |
| [#4, T_1 , P_A , ·, ·, #1] | [#10, T_3 , COMMIT, #8] |
| [#5, T_2 , P_C , ·, ·, #2] | [#11, T_2 , P_E , ·, ·, #5] |
| [#6, T_1 , P_E , ·, ·, #4] | |

Die angenommene Systemkonfiguration ist *no steal* ($\neg steal$) und *not force* ($\neg force$), außerdem wird das WAL-Prinzip eingehalten.

i) Kann ein Benutzer in dieser Situation bereits über den erfolgreichen Abschluss von T_3 informiert worden sein?

<input type="radio"/> ja <input type="radio"/> nein:

ii) Beschreiben Sie einen Ablauf (Änderungen an Log-Puffer und Log-Datei, evtl. neu zu erstellende Logeinträge oder nötige Auslagerungen anderer Seiten), der es erlaubt die Seite P_A in den persistenten Speicher auszulagern.

.....
.....
.....
.....

Aufgabe 3: Sperrprotokolle (Locking)

(12)

Ein wissenschaftliches Institut betreut sechs Experimente A, B, C, D, E, F , welche zwischen drei Projekten P_1, P_2 und P_3 geteilt werden. Jedes Projekt kann dabei den Wert eines Experiments E_i ablesen ($r(E_i)$), sowie die Parameter eines Experiments E_i verändern ($w(E_i)$). Projektenden sind durch c notiert.

Um sicherzustellen, dass es zu keiner Beeinflussung der Projekte untereinander kommt, wird der Zugriff auf die Experimente mittels dem Zwei-Phasen Sperrprotokoll mit wait-die Strategie (2PL mit wait-die) geregelt: Ablesen benötigt Share-Locks, verändern Exclusive-Locks. Sperren werden so spät wie möglich angefordert und so früh wie möglich wieder freigegeben. Als Zeitstempel wird das Jahr verwendet, in welchem das Projekt gestartet wurde.

a) Gegeben sind die folgenden Zugriffe der Projekte auf die Experimente:

$P_1(2019)$	b	$w(D)$	$r(A)$	$w(B)$	$w(E)$	$r(D)$	b	$r(D)$	c
$P_2(2020)$	b	$w(C)$				$r(B)$	o	$w(E)$	c
$P_3(2017)$	b		$w(F)$	$w(A)$	$w(E)$		t		
							$?$	$r(D)$	c

Aufgrund des Sperrprotokolls werden die Zugriffe nicht in der oben angegebenen Reihenfolge ausgeführt werden können. Geben Sie die tatsächliche Reihenfolge der Zugriffe an. Beim Eintrag $bot?$ soll ein Projekt neu gestartet werden, falls bis zu diesem Zeitpunkt ein Projekt wegen *wait-die* abgebrochen wurde. Muss ein Projekt auf eine Sperre warten, so werden alle "übersprungenen" Zugriffe nachgeholt, sobald die Sperre gewährt wurde.

	P_1	P_2	P_3
1		$w(C)$	
2	$w(D)$		
3	$r(A)$		
4			$w(F)$
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			

b) Kann es, unabhängig von a) – d.h. für eine beliebige Abfolge von Zugriffen, nicht nur der einen Sequenz aus a) – unter Einhaltung des Sperrprotokolls passieren, dass zwei Projekte P und Q Werte eines durch das jeweils andere Projekt veränderten Experiments ablesen?

Falls ja, geben Sie bitte eine entsprechende Folge von Zugriffen an.

Falls nein, begründen Sie kurz, warum dies nicht passieren kann.

ja nein

.....

.....

.....

.....

.....

.....

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

Aufgabe 4: Erstellen eines Datenbankschemas mittels SQL (7)

Folgendes Schema sei gegeben:

lehrende	(<u>name</u> , <u>typ</u> , haelt: <i>webinar.id</i>)
webinar	(<u>id</u> , titel, leitName: <i>lehrende.name</i> , leitTyp: <i>lehrende.typ</i>)
bedingungen	(<u>vor</u> : <i>webinar.id</i> , <u>nach</u> : <i>webinar.id</i>)
teilnehmende	(<u>kenn</u> , favorit: <i>webinar.id</i>)

Jede Lehrende ist durch ihren Namen und ihren Typ eindeutig gekennzeichnet. Der Typ muss ein Enum sein, bestehend aus den Werten “Univ-Ass”, “Ass-Prof” und “Univ-Prof”. Zusätzlich wird von jedem Lehrenden eines der Webinare, das von ihr gehalten wird, explizit vermerkt (**haelt**). Jedes Webinar ist eindeutig durch seine ID gekennzeichnet, darüber hinaus ist auch ein Titel vorhanden. Es wird auch gespeichert welche der Lehrenden, die dieses Webinar hält, als Leitende gilt. Da die Zahl 13 als unglückbringend gilt, soll kein Webinar die Zahl 13 oder ein Vielfaches davon als ID haben. Es gibt eine Relation Bedingungen, die vermerkt welche Webinare (**nach**) welche Anderen (**vor**) als Vorbedingung haben. Die Kombination aus Vorgänger- und Nachfolger-Webinar ist hier jeweils eindeutig. Zuletzt werden auch die Teilnehmenden gespeichert und sind über ihre Kennzeichnung eindeutig erkennbar. Alle Teilnehmenden haben ein Webinar als ihren Favorit bestimmt.

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

Hinweis: Achten Sie bei den Statements auf die Reihenfolge.

a) Erstellen Sie eine View `popular_webinar` basierend auf der folgenden Anfrage:

Gesucht sind die IDs aller "populären" Webinare, das sind Webinare welche von zumindest 10 Teilnehmern als Favorit vermerkt wurden.

Geben Sie die nötigen SQL Statements an, um diese View anzulegen.

b) Erstellen Sie folgende Rekursive SQL Abfrage:

Jede Zeile (`vor`, `nach`) der Tabelle `bedingungen` beschreibt eine *direkte Voraussetzung* zwischen Webinaren (`vor` ist eine direkte Voraussetzung von `nach`). Wir definieren ein Webinar als *Voraussetzung* eines anderen Webinars, falls es eine direkte Voraussetzung ist, oder die direkte Voraussetzung einer direkten Voraussetzung, usw. Als \mathcal{PW} (populäres Webinar) bezeichnen wir alle Webinare, die Favorit von mindestens 10 Teilnehmenden sind. Eine “ \mathcal{PW} -Voraussetzung” ist ein Webinar das sowohl eine Voraussetzung als auch ein \mathcal{PW} ist.

Gesucht sind sämtliche \mathcal{PW} -Voraussetzungen aller Webinare, welche von einer Lehrenden mit Namen “Diotima” und vom Typ “Univ-Prof” geleitet werden (das heißt, alle Webinare die bei `leitName` und `leitTyp` auf ebenjene Lehrende zeigen). Beachten Sie, dass diese selber **keine populären Webinare sein müssen**.

Die Abfrage soll die ID und den Titel der gesuchten Webinare ausgeben. Webinare die über mehrere “Pfade” als \mathcal{PW} -Voraussetzung auftauchen, sollen auch entsprechend mehrfach in der Liste auftauchen.

Geben Sie die nötigen SQL Statements an, um eine Rekursive Abfrage zu schreiben welche die beschriebene Abfrage implementiert. *Sie sind angehalten, die View aus dem vorherigen Beispiel hier wiederzuverwenden!*

Hinweis: Achten Sie darauf, dass ihre Abfrage terminiert. Sie dürfen jedoch davon ausgehen, dass die Datenbank keine zyklischen Voraussetzungen enthält.

Aufgabe 6: PL/SQL Trigger

(14)

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 Beispielinanz auf **Seite B**.

In jedem der folgenden tasks ist ein DELETE Statement gegeben das **über die Beispielinanz** ausgeführt wird. (Das DELETE 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)

```
DELETE FROM webinar WHERE id=6;
```

```
SELECT id FROM webinar;
```

```
SELECT * FROM bedingungen;
```

b)

```
DELETE FROM webinar WHERE id=1;
```

```
SELECT id FROM webinar;
```

```
SELECT * FROM bedingungen;
```

c)

```
DELETE FROM bedingungen WHERE nach=4 OR nach=3;
```

```
SELECT id FROM webinar;
```

```
SELECT * FROM bedingungen;
```

Gesamtpunkte: 70

Viel Erfolg!

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 delW() RETURNS TRIGGER AS $$
BEGIN
    IF OLD.leitTyp = 'Univ-Prof' THEN
        RETURN NULL;
    END IF;
    DELETE FROM Bedingungen WHERE vor = OLD.id;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trDw BEFORE delete ON webinar
FOR EACH ROW EXECUTE PROCEDURE delW();
```

```
CREATE FUNCTION delB() RETURNS TRIGGER AS $$
BEGIN
    IF (SELECT count(*) FROM webinar w
        JOIN teilnehmende t ON w.ID = t.favorit
WHERE w.id = OLD.nach) < 3
    THEN
        DELETE FROM webinar where ID = OLD.nach;
    END IF;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trDB after delete ON Bedingungen
FOR EACH ROW EXECUTE PROCEDURE delB();
```


Seite B
Beispielinstanz für Aufgabe 6:

Webinar

id	titel	leitName	leitTyp
1	Kreise und Bäume	A	Univ-Ass
2	HYPERgraphen	B	Ass-Prof
3	Digital Humanism	C	Univ-Prof
4	Dreiecke und Du	D	Ass-Prof
5	Epistemologie der Moral	C	Univ-Prof
6	Influencen	A	Univ-Ass
7	Unlogik	C	Univ-Ass
8	Logik	D	Ass-Prof

Bedingungen

vor	nach
1	2
1	3
3	4
2	4
4	5
6	7
7	8

Lehrende

name	typ	haelt
A	Univ-Ass	8
B	Ass-Prof	2
C	Univ-Prof	3
D	Ass-Prof	4
C	Univ-Ass	7

Teilnehmende

kenn	favorit
123	8
789	8
032	8
042	7
043	3
456	1
045	1
056	1

