Gruppe A

Bitte tragen Sie **sofort** und **leserlich** Namen, Studienkennzahl und Matrikelnummer ein und legen Sie Ihren Studentenausweis bereit.

PRÜFUNG AUS DATENBANKSYSTEME **VU 184.686**			11. 3. 2014
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 100 Minuten. Aufgaben sind auf den Angabeblättern zu lösen; Zusatzblätter werden nicht gewertet.

Aufgabe 1: (15)
Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

1. Bei den Einstellungen steal/force ist kein Undo möglich, aber ein Redo nötig. wahr ◯ falsch ⊗
2. Die Historie $r_1(A)$, $r_2(A)$, $w_1(A)$, $w_1(B)$, $w_2(B)$, c_1, c_2 ist nicht serialisierbar. wahr ⊗ falsch ◯
3. Eine Relation R sei an 6 Netzwerk-Knoten materialisiert mit den Gewichten von jeweils 10. Dann sind $Q_r(R) = 40$ und $Q_w(R) = 30$ gültige Lese- bzw. Schreib-Quoren. wahr ◯ falsch ⊗
4. Betrachten Sie drei Relationen $U(ABC)$, $V(ABD)$ und $W(CE)$. Dann gilt auf jeden Fall folgende Gleichheit: $((V \bowtie U) \bowtie W) = (U \rtimes (V \times W))$ wahr ⊗ falsch ◯
5. ACID garantiert die Eigenschaften *A*tomicity, *C*onsistency, *I*solation, und *D*urability. wahr ⊗ falsch ◯
6. Es gibt Relationen $R(\underline{A}B)$ mit 100 Tupeln und $S(AC)$ mit 100 Tupeln, für die der Ausdruck $R \ltimes S$ 1000 Tupeln ergibt. wahr ◯ falsch ⊗
7. Wenn ein Join mittels Hybrid Hash Join realisiert wird, dann kann das dadurch berechnete Ergebnis mehr Tupel umfassen, als wenn derselbe Join mittels Hash Join realisiert würde. wahr ◯ falsch ⊗
8. Mit PL/pgSQL lassen sich Anfragen formulieren, die man mit SQL-92 nicht formulieren könnte. wahr ⊗ falsch ◯
9. Die Anzahl der Zyklen im Wartegraphen entspricht immer der Anzahl der Transaktionen, die zurückgesetzt werden müssen, um einen Deadlock aufzulösen. wahr ◯ falsch ⊗
10. Die Gesamtkosten für die Sortierung einer Tabelle T sind $2b_T(1 + \lceil \log_{m-1}(\lceil b_T/m \rceil) \rceil)$, mit b_T als Anzahl der Seiten von T am Hintergrundspeicher und m als Anzahl der Seitenrahmen im Datenbankpuffer. wahr ⊗ falsch ◯

(Pro korrekter Antwort 1.5 Punkte, **pro inkorrekter Antwort -1.5 Punkte**, pro nicht beantworteter Frage 0 Punkte, für die gesamte Aufgabe mindestens 0 Punkte)

Aufgabe 2: (18)

Gegeben ist die folgende Historie von Transaktionen:

Schritt	T_1	T_2	T_3	Log: [LSN, TA, PageID, Redo, Undo, PrevLSN] or ⟨LSN, TA, PageID, Redo, PrevLSN, UndoNextLSN⟩
1	BOT			[#1, T_1, BOT, 0]
2		BOT		[#2, T_2, BOT, 0]
3			BOT	[#3, T_3, BOT, 0]
4	$r(C, c_1)$			
5		$r(C, c_2)$		
6			$r(A, a_3)$	
7		$w(A, c_2 * 2)$		[#4, T_2, P_A, A+=100, A-=100, #2] ....
8	$w(A, c_1 + 200)$			[#5, T_1, P_A, A+=100, A-=100, #1] ....
9			$r(B, b_3)$	
10			$w(B, a_3 + b_3)$	[#6, T_3, P_B, B+=100, B-=100, #3] ....
11		$w(C, c_2 + 200)$		[#7, T_2, P_C, C+=200, C-=200, #4] ....
12		$r(B, b_2)$		
13		$w(B, b_2 + c_2)$		[#8, T_2, P_B, B+=100, B-=100, #7] ....
14	commit			[#9, T_1, commit, #5] .....
15		commit		[#10, T_2, commit, #8] .....
16			rollback ...	[#11, T_3, rollback, #6] .....
17				⟨#12, T_3, P_B, B-=100, #11, #3⟩ .....
18				⟨#13, T_3, (BOT), #12, 0⟩ .....

(a) Nehmen Sie an, dass in Zeile 16 auch die Transaktion T_3 mittels commit beendet wird. Ist die resultierende Historie serialisierbar?

◯ ja ⊗ nein

Wenn ja, in Reihenfolge [T - ..... vor T - ..... vor T - .....].

Wenn nein: die Historie wird durch das Streichen von zumindest 1 ..... Operationen serialisierbar.

(b) Führen Sie nun – unabhängig davon, ob die Historie serialisierbar ist – in Zeile 16 ein *rollback* für T_3 durch.

Zu Beginn ist der relevante Datenbestand in der Datenbank $A = 100$, $B = 200$ und $C = 100$. Tragen Sie nun das Recovery-Log zu dieser Historie (mit rollback von T_3 in Zeile 16) in die rechte Spalte der obigen Tabelle ein. Dabei sind Undo/Redo-Einträge *relativ* zum Datenbestand anzugeben. Geben Sie in den Zeilen 16 ff. die Log-Einträge für die Recovery an.

Die Werte von A, B und C nach dem rollback von T_3 sind [A : 300 .....; B : 300 .....; C : 300 .....].

Aufgabe 3: Verteilte Datenbanken (12)

Zeigen Sie, dass die *Majority-Consensus* Methode zur Synchronisation replizierter Daten einen Spezialfall der *Quorum-Consensus*-Methode darstellt.

(a) Wie müssen die Gewichte Q_w und Q_r vergeben werden, um mit dem *Quorum-Consensus* Verfahren die *Majority-Consensus* Methode zu simulieren? [6]

Im einfachsten Fall wird jeder Kopie dasselbe Gewicht zugeordnet, das heißt zum Beispiel ein Gewicht von 1: $w_i = 1$ für $1 \leq i \leq n$. Damit ist das Gesamtgewicht $W(A) = n$.
Damit bei *Majority-Consensus* eine Transaktion ein Element A schreiben kann, müssen mehr als die Hälfte der Kopien beschrieben werden, das heißt $Q_w(A) = \lfloor n/2 \rfloor + 1$. Damit eine Transaktion ein Element A lesen kann, müssen mehr als die Hälfte der Kopien gelesen werden: $Q_r(A) = \lfloor n/2 \rfloor + 1$.

(b) Zeigen Sie, dass durch die Wahl der Quoren beide Bedingungen des Quorum-Consensus-Verfahrens erfüllt sind. [6]

1. $2 * Q_w(A) > W(A) \Longrightarrow 2 * (\lfloor n/2 \rfloor + 1) > n$
2. $Q_r(A) + Q_w(A) > W(A) \Longrightarrow 2 * (\lfloor n/2 \rfloor + 1) > n$

Die folgende Datenbankbeschreibung gilt für die Aufgaben 4 – 7:

Gegeben ist folgendes stark vereinfachtes Datenbankschema, in dem die Platzreservierung eines Kinos abgebildet wird.

`Saal`(sid, reihen, plaetze)
`Reservierung`(rid, vid, saal: *Saal.sid*, reihe, platz, gekauft)

Auf der letzten Seite dieser Prüfung finden Sie eine Beispielinstanz dieses Schemas!

In der Tabelle `Saal` werden die verschiedenen Säle des Kinos gespeichert. Das Attribut `reihen` speichert die Anzahl der Reihen, das Attribut `plaetze` die Plätze pro Reihe. Der Einfachheit halber gehen wir hier davon aus, dass in jeder Reihe gleich viele Plätze vorhanden sind.

In der Tabelle `Reservierung` werden die Platzreservierungen `rid` für die Vorstellungen `vid` gespeichert. Der Einfachheit halber bilden wir Vorstellungen nicht als separate Entitäten ab, sondern speichern den Saal der Vorstellung (Attribut `saal`) direkt. Das Attribut `reihe` speichert die Reihe des reservierten Platzes (ein ganzzahliger Wert, gezählt wird ab eins). Das Attribut `platz` speichert den Platz der Reservierung innerhalb der Reihe (ebenfalls ein ganzzahliger Wert ab eins). Das Attribut `gekauft` ist ein Boolscher Wert, der angibt ob die Karte nur reserviert ist (`false`) oder bereits gekauft ist (`true`).

Treffen Sie plausible Annahmen bezüglich der Datentypen der Attribute.

Aufgabe 4: (6)

Geben Sie CREATE TABLE Statements mit allen nötigen Constraints für die beiden Tabellen an.

```
CREATE TABLE Saal (
        sid VARCHAR(20) PRIMARY KEY,
        reihen INTEGER,
        plaetze INTEGER
);

CREATE TABLE Reservierung (
        bid INTEGER,
        vid INTEGER,
        saal VARCHAR(20) REFERENCES Saal(sid),
        reihe INTEGER,
        platz INTEGER,
        gekauft BOOLEAN,
        PRIMARY KEY (bid, vid)
);
```

Aufgabe 5: (6)

Evaluieren Sie die folgenden SQL-Statements bezüglich der Datenbankinstanz **reservierungen** (siehe letzte Seite), und geben Sie die Ausgaben der Abfrage an. Falls mehrere Ergebnisse zurückgegeben werden, trenne sie diese durch ; (Semikolons).

```
SELECT COUNT(*) FROM Reservierung, Saal;
```

30

```
SELECT COUNT(*) FROM Reservierung, Saal WHERE saal=sid;
```

10

```
SELECT COUNT(*) FROM Reservierung GROUP BY vid;
```

3 ; 1 ; 3 ; 3

```
SELECT COUNT(*) FROM Reservierung GROUP BY saal;
```

3 ; 1 ; 6

Aufgabe 6: (9)

Erstellen Sie einen PL/pgSQL Trigger `checkReservation`, der verhindert, dass Reservierungen hinzugefügt werden, die sich auf nicht vorhandene Plätze beziehen. Soll beispielsweise für Saal A (5 Reihen zu je 6 Plätzen) eine Reservierung für Reihe 3 Platz 17 hinzugefügt werden, soll eine Exception geworfen werden (denn Platz 17 existiert nicht). Gleiches gilt für Reihe 10 Platz 1 (denn Reihe 10 existiert nicht). Werfen Sie eine Exception mit dem Namen `illegalSeat`.

```
CREATE OR REPLACE FUNCTION fCheckReservation() RETURNS TRIGGER AS $$
DECLARE
   illegalSeat EXCEPTION;
   reihen INTEGER;
   plaetze INTEGER;
BEGIN
   SELECT reihen, plaetze INTO reihen, plaetze
      FROM Saal WHERE sid=NEW.saal
   IF (NEW.reihe < 1 OR NEW.reihe > reihen
       OR NEW.platz < 1 OR NEW.platz > plaetze) THEN
      RAISE EXCEPTION illegalSeat;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER checkReservation BEFORE UPDATE ON Reservierung FOR EACH ROW
EXECUTE PROCEDURE fCheckReservation();
```

Aufgabe 7: (9)

Vervollständigen Sie die Java Methode `auslastung`, der eine Vorstellung `vid` übergeben wird, und die

- die insgesamt im Saal der Vorstellung vorhandenen Reihen ausgibt
- für jede Reihe die Anzahl der gekauften Plätze ausgibt

Sie können davon ausgehen, dass für jede Vorstellung zumindest eine Reservierung vorliegt (das erlaubt, die Beziehung zwischen Vorstellung und Saal abzuleiten). Verwenden Sie ausschliesslich PreparedStatements (d.h. keine anderen Arten von Statements). Legen Sie die PreparedStatements hier an. Sie können hier auf eine Connection `c` zugreifen.

```
PreparedStatement psReihen = c.prepareStatement(
   "SELECT s.reihen FROM Saal s, Reservierung r WHERE r.saal = s.sid AND r.vid=?");
PreparedStatement psGekauft = c.prepareStatement(
   "SELECT COUNT(*) FROM Reservierung WHERE vid=? AND reihe=? AND bezahlt=true");
```

Vervollständigen Sie nun die Methode `auslastung`. Sie können die oben angelegten PreparedStatements verwenden. Um die genaue Formatierung der Ausgabe und die Fehlerbehandlung brauchen Sie sich nicht zu kümmern. Schliessen sie geöffnete Ressourcen (jedoch nicht solche, die bei der nächsten Ausführung der Methode noch benötigt werden).

```
public void auslastung(int vid) throws Exception {
   psReihen.setInt(1, vid);
   ResultSet rs = psReihen.executeQuery();
   rs.next();
   int reihen = psReihen.getInt(1);

   for (int reihe=1; reihe <= reihen; reihe++) {
      psGekauft.setInt(1, vid);
      psGekauft.setInt(2, reihe);
      ResultSet rs2 = psGekauft.executeQuery();
      rs2.next();
      System.out.println("Reihe " + reihe + ": " + rs2.getInt(1) + " Plätze gekauft.");
      rs2.close();
   }
   rs.close();
}
```

Gesamtpunkte: 75

Sie können diese Seite abtrennen und brauchen sie nicht abgeben!

Datenbankinstanz **reservierungen**:

Saal		
sid	reihen	plaetze
A	5	6
B	8	5
C	7	10

Reservierung					
rid	vid	saal	reihe	platz	gekauft
1	1	A	4	3	true
2	1	A	3	3	false
3	1	A	4	2	true
1	2	B	8	3	true
1	3	C	5	3	true
2	3	C	6	3	true
3	3	C	6	4	false
1	4	C	7	5	false
2	4	C	7	6	true
3	4	C	7	7	true