Übungsblatt 11

Aufgabe 11.1: statische Integritätsbedingungen (Aufgabe von E. Schicker)

Gegeben sei die folgende Relation zur Verwaltung des Personals eines Handelsunternehmens:

```
CREATE TABLE Mitarbeiter (
   PersNr INTEGER PRIMARY KEY,
   Vorname VARCHAR(30) NOT NULL,
   Nachname VARCHAR(30) NOT NULL,
   Strasse VARCHAR(30),
   PLZ INTEGER,
   Ort VARCHAR(20),
   GebDatum DATE NOT NULL CHECK (TIMESTAMPDIFF(YEAR, GebDatum, NOW()) BETWEEN 18 AND 65),
   Familienstand VARCHAR(11) NOT NULL CHECK (Familienstand IN ('ledig', 'verheiratet', 'geschieden', 'verwitwet')),
   Steuerklasse INTEGER CHECK (Steuerklasse IS NULL OR Steuerklasse BETWEEN 1 AND 6),
   Gehalt NUMERIC(8,2) NOT NULL CHECK (Gehalt BETWEEN 200 AND 10000)
);
```

Fügen Sie statische Integritätsbedingungen hinzu, die Folgendes sicherstellen:

- Das Gehalt beträgt 200 bis 10.000 (Euro).
- Der Familienstand ist „ledig“, „verheiratet“, „geschieden“ oder „verwitwet“.
- Die Steuerklasse ist entweder undefiniert (Minijob) oder ein Wert von 1 bis 6.
- Der Mitarbeiter ist mindestens 18 und höchstens 65 Jahre alt. Informieren Sie sich, welche Datumsfunktionen das Ihnen zur Verfügung stehende Datenbanksystem dafür anbietet.

Aufgabe 11.2: referentielle Integritätsbedingungen (Aufgabe von A. Kemper)

Wir betrachten das Universitätsschema, das mit sämtlichen Integritätsbedingungen angelegt worden ist und die bekannte Beispielausprägung hat (vgl. SQL-Relationen und Beispielausprägung in Moodle). Beschreiben Sie die Auswirkungen der folgenden Operationen:

- `DELETE FROM Vorlesungen WHERE Titel = 'Ethik';` Fehler und Rollback (Jonas hat sich in Ethik prüfen lassen)
- `INSERT INTO prüfen VALUES (24002, 5001, 2138, 2.0);` Fehler und Rollback (Prof 2138 gibt es nicht)
- `INSERT INTO prüfen VALUES (28106, 5001, 2127, 4.3);` Carnap hat in Grundzuege von Kopernikus eine 4.3 bekommen.
- `DROP TABLE Studenten;` Die Tabelle Studenten verschwindet, die Tabellen hören und prüfen verlieren alle Einträge

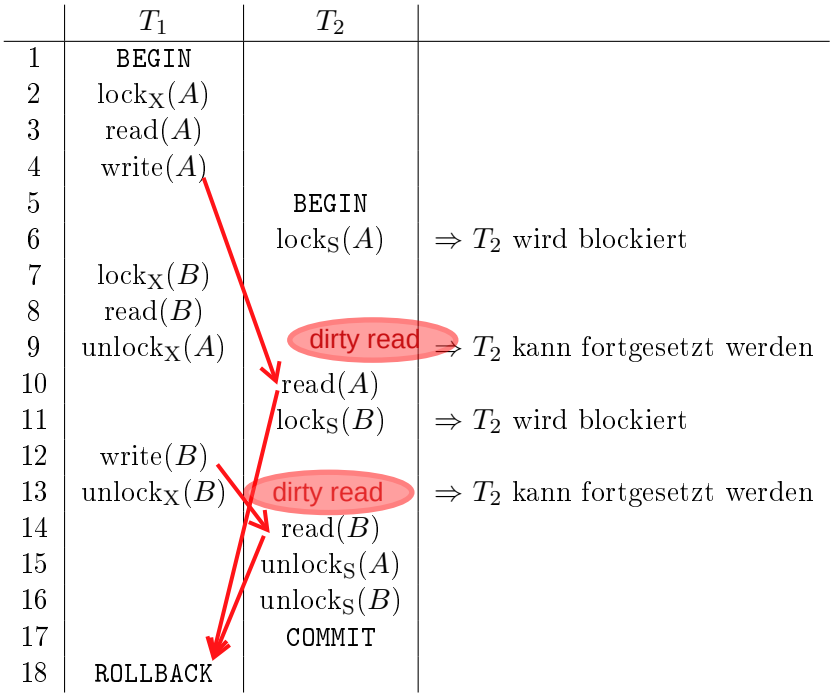
Aufgabe 11.3: Brute-Force-Deadlock-Vermeidung (Aufgabe von E. Schicker)

In einem kleinen Mehrbenutzer-Datenbanksystem existiere als Sperrmechanismus nur eine einzige Sperre auf die komplette Datenbasis. Kann in diesem System ein Deadlock entstehen? Warum ist ein derartiger Sperrmechanismus trotzdem nicht sinnvoll?

Die Datenbank arbeitet nur seriell. Dabei können keine Deadlocks entstehen.
Im Mehrbenutzerbetrieb ist die Datenbank langsam.

Aufgabe 11.4: Historienklassen

Betrachten Sie folgende Historie:



	T_1	T_2	
1	BEGIN		
2	$\text{lock}_\text{X}(A)$		
3	$\text{read}(A)$		
4	$\text{write}(A)$		
5		BEGIN	
6		$\text{lock}_\text{S}(A)$	$\Rightarrow T_2$ wird blockiert
7	$\text{lock}_\text{X}(B)$		
8	$\text{read}(B)$		
9	$\text{unlock}_\text{X}(A)$		$\Rightarrow T_2$ kann fortgesetzt werden
10		$\text{read}(A)$	
11		$\text{lock}_\text{S}(B)$	$\Rightarrow T_2$ wird blockiert
12	$\text{write}(B)$		
13	$\text{unlock}_\text{X}(B)$		$\Rightarrow T_2$ kann fortgesetzt werden
14		$\text{read}(B)$	
15		$\text{unlock}_\text{S}(A)$	
16		$\text{unlock}_\text{S}(B)$	
17		COMMIT	
18	ROLLBACK		

a) Welches der vier typischen Probleme des Mehrbenutzerbetriebs (Lost Update, Dirty Read, Fuzzy Read, Phantom Read) tritt hier auf?

b) Ist diese Historie serialisierbar? Ist sie rücksetzbar? Nicht serialisierbar (wegen Dirty Read), Nicht rücksetzbar (liest von T1 und committed vor T1)

c) Wie würde eine möglichst ähnliche Historie aussehen, die die beiden in b) genannten Eigenschaften besitzt? T1 commitet.

d) Welche Historie würde das strenge Zwei-Phasen-Sperrprotokoll bei erfolgreichem Abschluss von T_1 erzeugen und welche Eigenschaften würde diese besitzen? => Excel

Aufgabe 11.5: Synchronisationsverfahren

Es laufen zwei Transaktionen, T_1 und T_2. Ohne Synchronisation passiert dabei Folgendes:

1. T_1 startet und liest das Datum A.
2. T_1 schreibt das Datum A.
3. T_2 startet und liest das Datum A.
4. T_2 liest das Datum B und endet.
5. T_1 liest das Datum B.
6. T_1 schreibt das Datum B und endet.

Welche Historien erzeugen das strenge Zwei-Phasen-Sperrprotokoll, die Zeitstempel-basierte Synchronisation und die optimistische Synchronisation?

Blatt11Aufgabe5

strenge Zwei-Phasen-Sperrprotokoll	
T1	T2
BEGIN	
LockX (A)	
Read (A)	
Write (A)	
	BEGIN
	LockS (A) (wait)
LockX (B)	
Read (B)	
Write (B)	
COMMIT	
	Read (A)
	Read (B)
	COMMIT

Zeitstempelbasierte Synchronisation					
T1	T2	A.r	A.w	B.r	B.w
BEGIN		0	0	0	0
Read (A)		1			
Write (A)			1		
	BEGIN				
	Read (A)	2			
	Read (B)			2	
	COMMIT				
Read (B)				ok	
Write (B)				fail	
ROLLBACK					

optimistische Synchronisation	
T1	T2
BEGIN	
Read (A)	
Save (A)	
	BEGIN
	Read (A)
	Read (B)
	VALIDATE
	COMMIT
Read (B)	
Save (B)	
VALIDATE	
COMMIT A,B	

Blatt 11 Aufgabe 4	
strenge Zwei-Phasen-Sperrprotokoll	
T1	T2
BEGIN	
LockX (A)	
read (A)	
write (A)	
	BEGIN
	LockS (A)
LockX (B)	
read (B)	
write (B)	
ROLLBACK	
	read (A)
	LockS (B)
	read (B)
	COMMIT