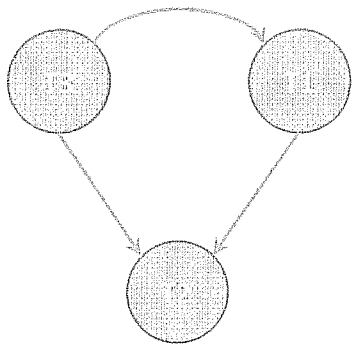
FernUniversität in Hagen

Lösungsvorschläge Leistungsnachweisklausur „1672 Datenbanken II“

31. August 2013

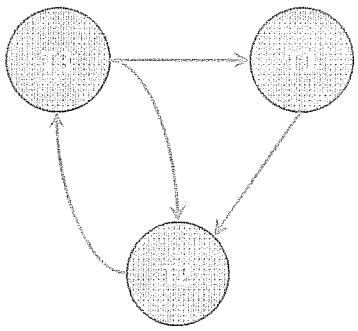
Aufgabe 1: Serialisierbarkeit — 12 Punkte

a)



Da der Graph keinen Zyklus enthält, ist die Schedule serialisierbar. Eine äquivalente serielle Schedule ist T3, T1, T2.

b)



Da der Graph einen Zyklus enthält, ist die Schedule nicht serialisierbar.

c)

Im Fall b) liegt die Ursache der Nicht-Serialisierbarkeit in der Abhängigkeit T2 → T3. Diese Abhängigkeit entsteht dadurch, dass T2 das Objekt Y liest, bevor es von T3 gelesen und geschrieben wird. Wenn wir diese Leseoperation weiter nach hinten ziehen (also verzögern), ist die Schedule serialisierbar:

Ursprüngliche Schedule:

S_2: $r_1(X)$; $r_2(Z)$; $r_3(X)$; $r_1(Z)$; $\underline{r_2(Y);}$ $r_3(Y)$; $w_1(X)$; $w_2(Z)$; $w_3(Y)$; $w_2(Y)$

Neue Schedule:

S_2: $r_1(X)$; $r_2(Z)$; $r_3(X)$; $r_1(Z)$; $r_3(Y)$; $w_1(X)$; $w_2(Z)$; $w_3(Y)$; $\underline{r_2(Y);}$ $w_2(Y)$

(pro Teilaufgabe 4 Punkte)

Aufgabe 2 (Multiple Choice) (10 Punkte)

W	F	
☒	☐	Ein mögliches Verfahren zur Einhaltung der Serialisierbarkeit ist das Zweiphasen-Sperrprotokoll.
☐	☒	Wenn garantiert ist, dass beim Commit einer Transaktion alle ihre Änderungen auf die Platte geschrieben werden, ist kein Undo-Log notwendig.
☐	☒	Ein fortgepflanztes Rollback von Transaktionen kann dadurch verhindert werden, dass alle Transaktionen ihre Sperren bereits zu Beginn der Transaktion erwerben.
☒	☐	Der Abhängigkeitsgraph zeigt an, welche Transaktion mit einer anderen in einem Zugriffskonflikt steht. Das heißt, eine Kante von T1 nach T2 in diesem Graphen zeigt an, dass T1 vor T2 auf ein Objekt zugreift und dass (mindestens) eine der beiden Transaktionen dabei eine Schreiboperation durchführt.
☒	☐	Ein Zyklus im Wartet-auf-Graphen zeigt an, dass die beteiligten Transaktionen in einem Deadlock verstrickt sind.
☒	☐	Wenn der Abhängigkeitsgraph keinen Zyklus hat, dann ist das Transaktionssystem serialisierbar.
☐	☒	Wenn der Abhängigkeitsgraph einen Zyklus hat, dann ist das Transaktionssystem nicht serialisierbar. *Dies stimmt zwar meistens, ein Gegenbeispiel dazu findet sich aber in KE2, S. 11.*
☐	☒	Intentionssperren auf einem Tupel zeigen an, dass eine Transaktion beabsichtigt, auf dieses Tupel eine Schreibsperre zu setzen.
☐	☒	Ein Hotspot ist ein Datenbereich, auf den viele Transaktionen zugreifen müssen. Deshalb muss dieser Datenbereich nach jeder Änderung aus dem Hauptspeicher auf die Platte geschrieben werden.
☒	☐	Trigger sind Programme oder Operationen, die vom Datenbanksystem beim Eintreten von bestimmten Bedingungen ausgeführt werden.

Aufgabe 3: Recovery (13 Punkte)

a) Ein physisches Log enthält die Before- und After-Images jeder Update-Operation. Ein logisches Log enthält nur einen Eintrag, der die Update-Operation beschreibt (etwa: „insert record r into file x"). (1 Punkt)

Vorteil: Der Speicheraufwand für das Log und die Belastung der I/O-Kanäle ist geringer, da beim physischen Log für eine einzige Operation u.U. eine ganze Reihe von Before- und After-Images geschrieben werden müssen. (1 Punkt)

Nachteil: Zum einen muss der Recovery-Manager erheblich erweitert werden, so dass er in der Lage ist, nicht nur die entsprechenden Operationen „nachzuführen", sondern auch die inversen Operationen ausführen kann. Zum anderen ist es für die UNDO-Operation wesentlich zu wissen, ob die entsprechende Operation sich bereits auf die Datenbank ausgewirkt hat oder nicht (unerheblich für das UNDO beim physischen Log.) (2 Punkte)

b) Beim Shadow-Verfahren werden alle Änderungen zum Commit-Zeitpunkt in einer einzigen atomaren Operation in die Datenbank übertragen. Dazu wird auf Kopien der zu verändernden Objekte gearbeitet, die über Hilfskataloge verwaltet werden. Zum Commit-Zeitpunkt tauschen der Haupt- und der Hilfskatalog ihre Rollen. (1,5 Punkte)

Das Verfahren wird für die Realisierung der NO-UNDO/NO-REDO Strategie eingesetzt. Folglich muss auch im Falle eines Systemabsturzes nichts zur Wiederherstellung der Konsistenz geschehen, da ja nur auf Kopien gearbeitet wurde. (1,5 Punkte)

c) In diesem Fall ist die UNDO/REDO Recovery-Strategie anzuwenden, d.h., das Recovery muss folgendermaßen aussehen: (2 Punkte)

T1 und T5 müssen nicht berücksichtigt werden, da sie zum Zeitpunkt des Checkpoints bereits abgeschlossen waren und damit auf der externen Platte gespeichert sind. (1,5 Punkte)
T2 und T4 müssen zurückgesetzt werden (Rollback), d.h. die Before-Images müssen in die Datenbank eingebracht werden. (1,5 Punkte)
T3 muss wiederholt werden, d.h. die After Images werden in die Datenbank eingebracht. (1 Punkt)

Aufgabe 4: Hierarchische Sperren (16 Punkte)

Im Folgenden arbeiten wir mit den folgenden Kürzeln:

T1, T2, T3: Transaktionen
K1-K15: Knoten (siehe Graphik in der Aufgabenstellung)

a) Wir benötigen die folgenden Sperrtypen: (4 Punkte)
- W Schreibsperre (write lock)
- R Lesesperre (read lock)
- IW Intentionssperre zum Schreiben
- IR Intentionssperre zum Lesen

Die Kompatibilitätsmatrix sieht folgendermaßen aus:

	W	R	IW	IR
W	Nein	Nein	Nein	Nein
R	Nein	Ja	Nein	Ja
IW	Nein	Nein	Ja	Ja
IR	Nein	Ja	Ja	Ja

b) T1 möchte das Tupel 11 mit einer Schreibsperre (W) versehen.

(T1, K1, IW)
(T1, K2, IW)
(T1, K5, IW)
(T1, K11, W)

c) T1 möchte die Relation 4 zum Schreiben sperren.

(T1, K1, IW)
(T1, K2, IW)
(T1, K4, W)

d) T2 möchte das Tupel 10 zum Lesen sperren.

(T2, K1, IR)
(T2, K2, IR)
(T2, K5, IR)
(T2, K10, R)

e) T3 möchte das Tupel 9 zum Schreiben sperren.

(T3, K1, IW)
(T3, K2, IW)
(T3, K4, IW) Diese Sperre wird nicht gewährt, da sie in Konflikt zu der in c) gewährten Schreib-Sperre steht.

f) T2 möchte das Tupel 12 zum Lesen sperren.

(T2, K1, IR)
(T2, K3, IR)
(T2, K6, IR)
(T2, K12, R)

g) T1 möchte die ganze Relation 6 zum Schreiben sperren.

(T1, K1, IW)
(T1, K3, IW)
(T1, K6, W) Diese Sperre wird nicht gewährt, da sie in Konflikt zu der in f) gegebenen IS Sperre auf K6 steht.

(b) – g) jeweils 2 Punkte)

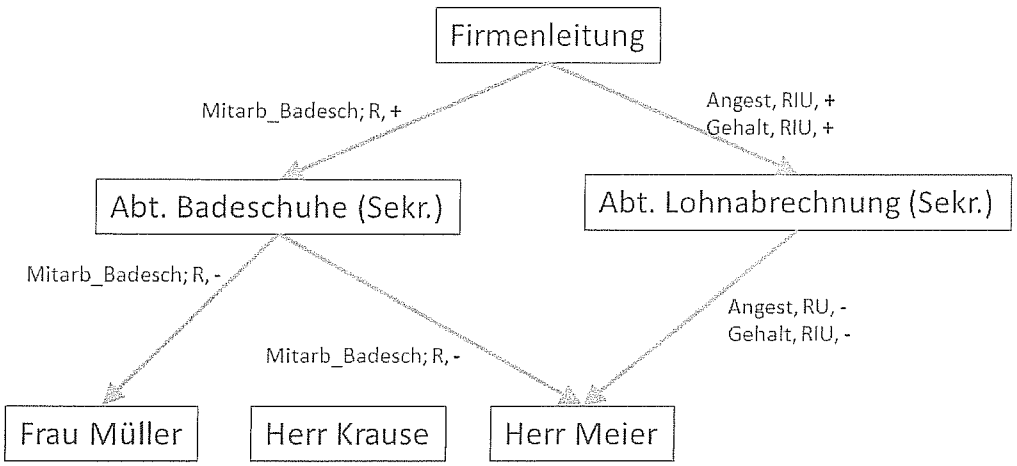
Aufgabe 5: Zugriffskontrolle (12 Punkte)

a) Um sicherzustellen, dass in einer Fachabteilung beim Zugriff auf die Tabelle „Mitarbeiter“ nur die Daten der zur Abteilung gehörenden Mitarbeitern gesehen werden können, wird zunächst eine View definiert (Mitarbeiter_ABT_X), die in einer Selektionsbedingung die Mitarbeiter-Sätze der Abteilung auswählt. In SQL könnte das in etwa folgendermaßen aussehen:

```
CREATE VIEW Mitarbeiter_Badeschuhe as
   (     SELECT     *
         FROM       Mitarbeiter
         WHERE      Mitarbeiter.Abt = 'Badeschuhe'
   )
```

Anschließend kann das Recht, auf diese View lesend zuzugreifen, per GRANT-Befehl weitergegeben werden (siehe Aufgabe b)) (3 Punkte)

b) Graphische Darstellung (3 Punkte):



Kanten-Bezeichnungen: <Tabelle, Rechte, mit Weitergabe?>
Beispiel: <Angest,RIU,+>: Für die Tabelle Angest werden die Rechte zum Lesen, Einfügen und Ändern weitergegeben.

SQL Befehle: (3 Punkte)

Firmenleitung:

```
GRANT select ON Mitarbeiter_Badeschuhe
      TO Badeschuhe WITH GRANT OPTION
GRANT select, update, insert ON Gehalt
      TO Lohnabrechnung WITH GRANT OPTION
GRANT select, update, insert ON Angestellte
      TO Lohnabrechnung WITH GRANT OPTION
```

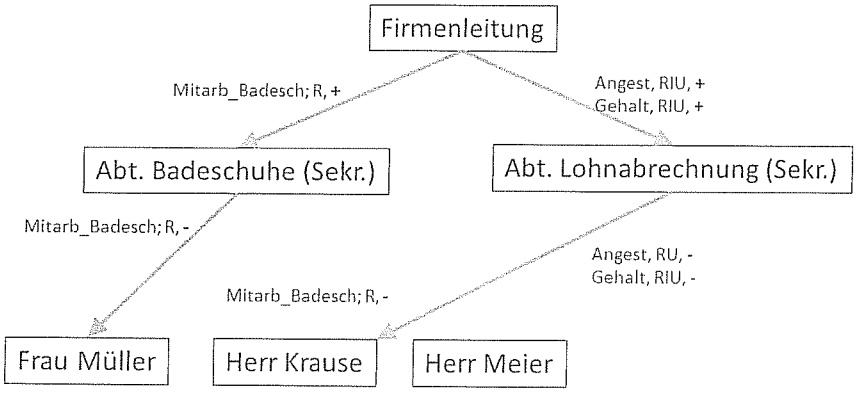
Badeschuhe:

```
GRANT select ON Mitarbeiter_Badeschuhe TO Müller
/
GRANT select ON Mitarbeiter_Badeschuhe TO Meier
/
```

Lohnabrechnung:

```
GRANT select, update, insert ON Gehalt
        TO Lohnabrechnung WITH GRANT OPTION
GRANT select, update, insert ON Angestellte
        TO Lohnabrechnung WITH GRANT OPTION
```

c) Herr Meier verliert die von der Abteilung Lohnabrechnung vergebenen Rechte. Diese Rechte werden von der Abteilung Lohnabrechnung nun an Herrn Krause übergeben. Ebenso verliert Herr Meier die Leserechte an der View Mitarbeiter_Badeschuhe.



Lohnabrechnung:

```
REVOKE select, update, insert ON Gehalt FROM Meier
GRANT select, update, insert ON Gehalt TO Krause
```

Badeschuhe

```
REVOKE select ON Mitarbeiter_Badeschuhe FROM Meier
```

(3 Punkte)

Aufgabe 6: Erweiterungen des relationalen Modells (12 Punkte)

a)

T_NR	T_Name	Breite	Höhe	Variante
				Länge

T_Nr	T_Name	Breite	Höhe	Variante
2	Serie 1	60	70	Länge: 100, 120, 160
3	Serie 2	80	70	Länge: 120, 160, 200

b)

```
CREATE TYPE Varianten_Type AS
    (       Länge       INTEGER,
            Material    VARCHAR (100)
    )

CREATE TYPE Tisch_Produkte AS
    (       T_NR        INTEGER,
            T_Name      VARCHAR (100),
            Breite      INTEGER,
            Höhe        INTEGER,
            Variante    Varianten_Type multiset
    )

CREATE TABLE Tisch_Tabelle of Tisch_Produkte
```

Alternativ statt der beiden letzten CREATE-Statements:

```
CREATE TABLE Tisch_Tabelle AS
    (       T_NR        INTEGER,
            T_Name      VARCHAR (100),
            Breite      INTEGER,
            Höhe        INTEGER,
            Variante    Varianten_Type multiset
    )
```