TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

# Übung zur Vorlesung
## *Einsatz und Realisierung von Datenbanksystemen* im SoSe14

Moritz Kaufmann (moritz.kaufmann@tum.de)
http://www-db.in.tum.de/teaching/ss14/impldb/

**Blatt Nr. 2**

### Aufgabe 1

1. Geben Sie alle Eigenschaften an, die von der Historie erfüllt werden.

$$w_1(x), r_2(y), w_3(y), w_2(x), w_3(z), c_3, w_1(z), c_2, c_1$$

2. Geben Sie alle Eigenschaften an, die von der Historie erfüllt werden.

$$r_1(x), r_1(y), w_2(x), w_3(y), r_3(x), a_1, r_2(x), r_2(y), c_2, c_3$$

3. Gegeben die unvollständige Historie:

$$H = w_1(x), w_1(y), r_2(x), r_2(y)$$

   a) Fügen Sie `commit`s in $H$ so ein, dass die Historie RC aber nicht ACA erfüllt:

   b) Fügen Sie `commit`s in das ursprüngliche $H$ so ein, dass die Historie ACA erfüllt.

### Aufgabe 2

Wäre es beim strengen 2PL-Protokoll ausreichend, alle Schreibsperren bis zum EOT zu halten, aber Lesesperren schon früher wieder abzutreten? Begründen Sie Ihre Antwort.

### Aufgabe 3

Weisen Sie (halbwegs) formal nach, dass das 2PL-Protokoll nur serialisierbare Historien zulässt.

### Aufgabe 4

Skizzieren Sie einen Ablauf von Transaktionen, bei dem ein Deadlock auftritt, der einen Zyklus mit einer Länge von mindestens 3 Kanten im Wartegraphen erzeugt.

### Aufgabe 5

Nennen Sie die Vorteile und Nachteile von Deadlockerkennung / Vermeidung durch:

- Timeouts
- Wartegraphen
- Preclaiming
- Zeitstempel

Sind Kombinationen denkbar/sinnvoll?

## Aufgabe 6

Beim „multiple-granularity locking“ (MGL) werden Sperren von oben nach unten (top-down) in der Datenhierarchie erworben. Zeigen Sie mögliche Fehlerzustände, die eintreten könnten, wenn man die Sperren in umgekehrter Reihenfolge (also bottom-up) setzen würde.

## Gruppenaufgabe (Muss nicht zuhause vorbereitet werden.)

Gegeben die Relation „Aerzte“, die den Bereitschaftsstatus von Ärtzen modelliert

| Name | Vorname | ... | Bereit |
|---|---|---|---|
| House | Gregory | ... | ja |
| Green | Mark | ... | nein |
| Brinkmann | Klaus | ... | ja |

sowie die folgende Transaktion in Pseudocode:

```
dienstende(arzt_name)
  select count(*) into anzahl_bereit from aerzte where bereit='ja'
  if anzahl_bereit > 1 then
    update aerzte set bereit='nein' where name=arzt_name
```

Die Transaktion soll dafür sorgen, dass immer mindestens ein Arzt bereit ist.

Betrachten Sie einen Ablauf, bei dem zwei zur Zeit bereite Ärzte zum gleichen Zeitpunkt entscheiden, ihren Status auf „nein “, d.h. nicht bereit zu ändern:

$T_1$: execute dienstende('House')
$T_2$: execute dienstende('Brinkmann')

Gehen Sie beispielsweise davon aus, dass das DBMS versucht, die Transaktion jeweils abwechselnd zeilenweise abzuarbeiten.

Diskutieren Sie:

- Was kann bei Snapshot Isolation passieren?
- Warum ist dies bei optimistischer Synchronisation nicht möglich?
- Wie verhält sich strenges 2PL?