Tutorium 11 - Transaktionen

25.01.2023 – Finn Kapitza

1. Aufgabe

Aufgabe 11.1 - Synthesealgorithmus

AssistentProfessorDiplomand (

PersNr,	$\leftarrow$	*Personalnummer des Assistenten*
Name,	$\leftarrow$	*Name des Assistenten*
Fachgebiet,	$\leftarrow$	*Fachgebiet des Assistenten*
ChefPersNr,	$\leftarrow$	*Personalnummer des Professors*
ChefName,	$\leftarrow$	*Name des Professors*
MatrNr,	$\leftarrow$	*Matrikelnummer des Studenten*
StudName,	$\leftarrow$	*Name des Studenten*
Semester,	$\leftarrow$	*Fachsemester des Studenten*
StudWohnOrt	$\leftarrow$	*Wohnort des Studenten*

Enthält Daten von Studenten, deren Diplomarbeit von Assistenten betreut wird, die von Professoren angestellt sind

- ChefPersNr $\rightarrow$ ChefName
- PersNr $\rightarrow$ Name, Fachgebiet, ChefPersNr, ChefName
- MatrNr $\rightarrow$ PersNr, Name, Fachgebiet, ChefPersNr, ChefName, StudName, Semester, StudWohnOrt

Aufgabe 11.1.a – Bestimme alle Schlüsselkandidaten

- ChefPersNr $\rightarrow$ ChefName
- PersNr $\rightarrow$ Name, Fachgebiet, ChefPersNr, ~~ChefName~~
- MatrNr $\rightarrow$ PersNr, ~~Name, Fachgebiet, ChefPersNr, ChefName~~, StudName, Semester, StudWohnOrt

- Nur die Attribute auf den linken Seiten können in Schlüsselkandidaten enthalten sein -> *ChefPersNr*, *PersNr* und *MatrNr*

- Weder *ChefPersNr* noch *PersNr* sind Eindeutig

- *MatrNr* ist der einzige Schlüsselkandidat

Aufgabe 11.1.b – Synthesealgorithmus

1) Kanonische Überdeckung:
   a) **Linksreduktion**:
      Da alle funktionalen Abhängigkeiten nur ein Attribut enthalten, kann nichts reduziert werden
   b) **Rechtsreduktion**:

- $PersNr \rightarrow Name, Fachgebiet, ChefPersNr$, da:
  $$ChefName \in AttrHülle(F - (PersNr \rightarrow Name, Fachgebiet, ChefPersNr, ChefName) \cup (PersNr \rightarrow Name, Fachgebiet, ChefPersNr, \{PersNr\})$$

- $MatrNr \rightarrow StudName, Semester, StudWohnOrt, PersNr$, da:
  $\{Name, Fachgebiet, ChefPersNr, ChefName\} \subseteq$
  $AttrHülle(F -$
  $(MatrNr \rightarrow StudName, Semester, StudWohnOrt, PersNr, Name, Fachgebiet, ChefPersNr, ChefName)$
  $\cup (MatrNr \rightarrow StudName, Semester, StudWohnOrt, PersNr), \{MatrNr\})$

Aufgabe 11.1.b - Synthesealgorithmus

$$F = \{$$
$$ChefPersNr \rightarrow ChefName,$$
$$PersNr \rightarrow Name, Fachgebiet, ChefPersNr,$$
$$MatrNr \rightarrow StudName, Semester, StudWohnOrt, PersNr\} = F_c$$

c) **Entfernung rechtsleerer Abhängigkeiten**: nichts zu tun

d) **Zusammenfassen gleicher linker Seiten**: nichts zu tun

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 11.1.b - Synthesealgorithmus

2) Erstellen eines neuen Relationenschemas aus F_c:

- **Professor**(<u>ChefPersNr</u>, ChefName)
- **Assistent**(<u>PersNr</u>, Fachgebiet, Name, ChefPersNr)
- **Diplomand**(<u>MatrNr</u>, StudName, Semester, StudWohnOrt, PersNr)

3) Rekonstruktion eines Schlüsselkandidaten:

-> Diplomand enthält Schlüsselkandidaten (*MatrNr*)

4) Elimination überflüssiger Relationen:

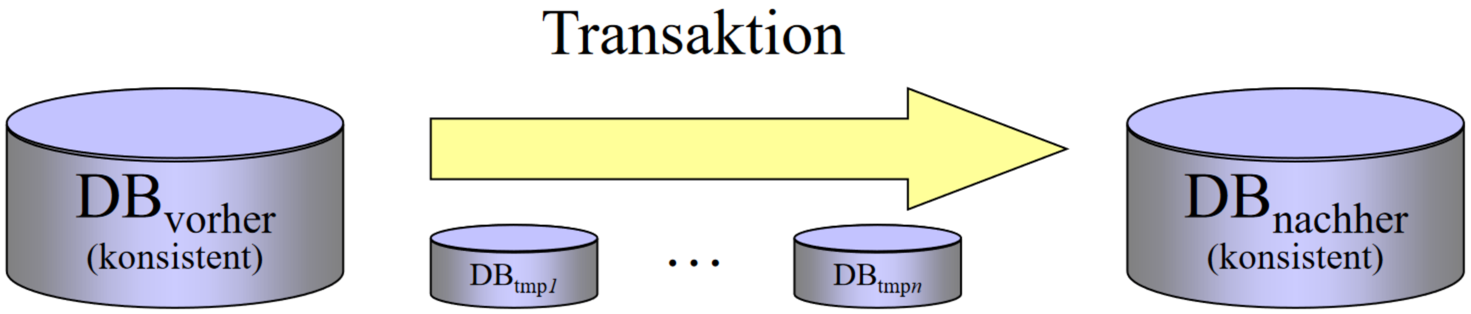
-> nichts zu tun

2. Wiederholung – Transaktionen

Was sind Transaktionen?

Transaktion: Folge von Befehlen (*read*, *write*), die die Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt

-> Einheit integritätserhaltender Zustandsänderung einer Datenbank



Transaktionen (TA)

Hauptaufgaben von Transaktionen:

- Synchronisation (Koordination mehrerer Benutzerprozesse)
- Recovery (Beheben von Fehlersituationen)

Eigenschaften von Transaktionen:

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

Warum Transaktionen?

- Datenbanken werden selten von nur einem Benutzer benutzt
  -> mehrere Nutzer arbeiten **gleichzeitig** auf den Daten

- Die Nutzer sollen nichts von den anderen Nutzern mitbekommen
  -> **logischer Einnutzerbetrieb**

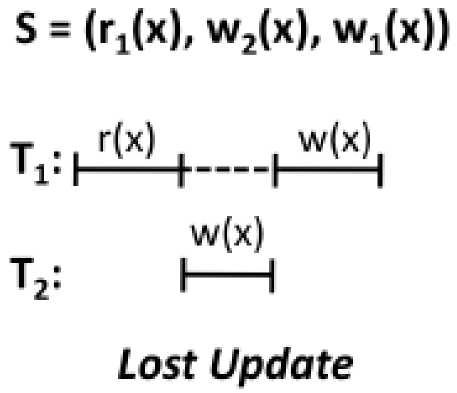
- Pseudoparallele Ausführung der Transaktionen -> bessere Auslastung des Systems

- Ohne Kontrolle des Ablaufs von TAs -> **Anomalien** können auftreten

Anomalien – Lost Update

- Verloren gegangene Änderung
- Zwei Transaktionen: T_1, T_2

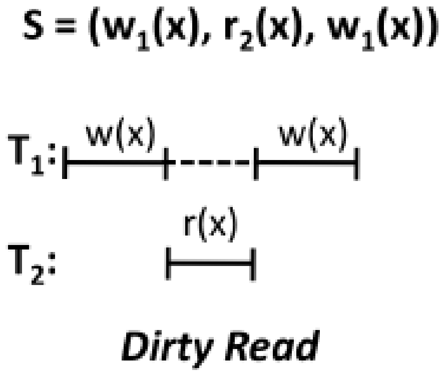
- T_2 ändert Objekt x -> Änderung wird durch T_1 überschriebe



$\Rightarrow$ Änderung von T_2 geht verloren

$\Rightarrow$ Verstoß gegen **Durability**

Anomalien – Dirty Read / Dirty Write



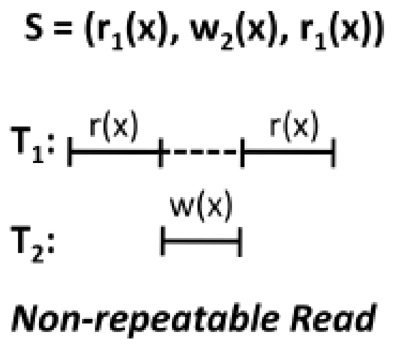
- Zugriff auf „schmutzige“ (nicht dauerhaft gültige) Daten
- Zwei Transaktionen: T_1, T_2

- T_1 verändert Objekt x zwei mal
- Zwischen den Änderungen liest T_2 den Wert von x (dieser Wert bleibt aber nicht dauerhaft gültig)

$\Rightarrow$ Verstoß gegen **Consistency**

Anomalien – Non-Repeatable Read

- Eine Transaktion sieht während ihrer Ausführung zwei unterschiedliche Werte von einem Objekt



- T_1 liest beim ersten Auslesen von x einen anderen Wert als beim zweiten, da T_2 den Wert von x verändert

⇒ Verstoß gegen **Isolation**

⇒ Transaktionen müssen so Ausgeführt werden, dass das **ACID**-Prinzip nicht verletzt wird

Schedules

Ein **Schedule** S ist eine Folge von Aktionen für eine Menge $T = \{T_1, \ldots T_n\}$ von Transaktionen.

S entsteht durch Mischen der Aktionen der TA T_i
-> Reihenfolge der Aktionen innerhalb der jeweiligen TA wird beibehalten

Serieller Schedule:

Ein Schedule ist **seriell**, wenn die einzelnen Transaktionen $T = \{T_1, \ldots T_n\}$ nicht verzahnt werden, sondern blockweise hintereinander ausgeführt werden

=> **Isolation** ist gegeben, aber langsam, da nicht parallel

Serialisierbarer Schedule:

Schedule S von $T = \{T_1, \ldots T_n\}$ ist **serialisierbar**, wenn er die gleiche Wirkung hat wie ein beliebiger **serieller** Schedule von $T = \{T_1, \ldots T_n\}$

$\Rightarrow$ Auch hier ist die **Isolation** nicht verletzt

$\Rightarrow$ Bessere Performanz als bei **seriellen** Schedules

$\Rightarrow$ Nur **serialisierbare** Schedules dürfen zugelassen werden

Abhängigkeiten in Schedules

Abhängigkeit: unterschiedliche Transkationen, gleiches Objekt

Sei S ein Schedule:

- Schreib-Lese Abhängigkeit von $T_i \rightarrow T_j$:
  - Es gibt ein x, so dass in S $\mathrm{w_i}(x)$ vor $\mathrm{r_j}(x)$ kommt -> $\mathrm{wr_{i,j}}(x)$

- Lese-Schreib Abhängigkeit von $T_i \rightarrow T_j$:
  - Es gibt ein x, so dass in S $r_i(x)$ vor $w_j(x)$ kommt -> $rw_{i,j}(x)$

- Schreib-Schreib Abhängigkeit von $T_i \rightarrow T_j$:
  - Es gibt ein x, so dass in S $w_i(x)$ vor $w_j(x)$ kommt -> $ww_{i,j}(x)$

Serialisierungsgraph

- Ein Serialisierungsgraph G ist ein gerichteter Graph $G = (V, E)$:
  - Mit Knoten $V = \{T_1, \dots, T_n\}$ (beteiligten Transaktionen)
  - Mit Kanten E = Abhängigkeiten $(T_i \rightarrow T_j)$ zwischen den Transaktionen

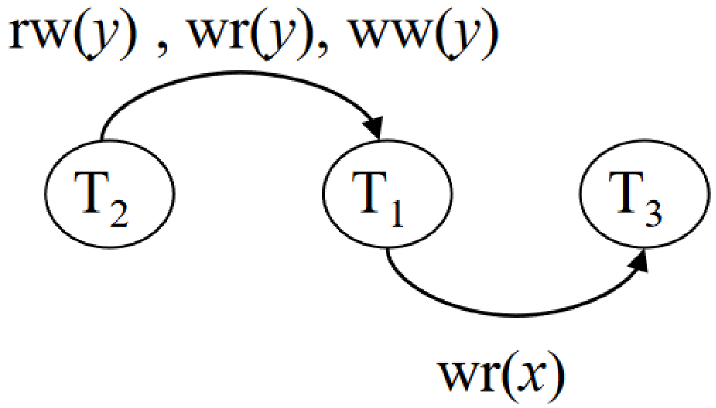
- Wenn der Graph zyklenfrei ist -> serialisierbar

- Wenn der Graph Zyklen enthält -> nicht serialisierbar

Serialisierungsgraph - Beispiel

$S = (r_1(x), r_2(y), r_3(z), w_3(z), w_2(y), w_1(x), w_2(y), r_1(y), r_3(x), w_1(y))$

- Abhängigkeiten: $wr_{1,3}(x), rw_{2,1}(y), wr_{2,1}(y), ww_{2,1}(y)$



$\Rightarrow$ Zyklenfrei -> S ist serialisierbar

Aufgabe 11.2 – Kombinatorik von Schedules

Gegeben sei eine Menge von n Transaktionen $\{T_1, ..., T_n\}$, wobei jede Transaktion T_i jeweils aus i_n vielen Einzeloperationen $T_i = \langle A_{i,1}, A_{i,2}, ..., A_{i,i_n} \rangle$ besteht.

Beispiel:
$T_1 = \langle A_{1,1}, A_{1,2}, A_{1,3}, A_{1,4} \rangle$
$T_2 = \langle A_{2,1}, A_{2,2}, A_{2,3} \rangle$
$T_3 = \langle A_{3,1}, A_{3,2}, A_{3,3} \rangle$

Erläutern Sie für das Beispiel $\{T_1, T_2, T_3\}$ sowie für den allgemeinen Fall $\{T_1, \ldots, T_n\}$:

Aufgabe 11.2.a – Anzahl beliebiger Schedules

Aktionen dürfen innerhalb der Transaktion nicht vertauscht werden -> sonst beliebig angeordnet

Im Beispiel:

- Es gibt 10 Aktionen -> 10 freie Plätze im Schedule
- Erste Transaktion bekommt 4 beliebige Plätze
- Zweite TA bekommt 3 beliebige aus den übrigen 10-4 = 6 Plätzen
- Dritte TA bekommt den Rest

$$\binom{10}{4} \cdot \binom{6}{3} \cdot \binom{3}{3} = \frac{10!}{4! \cdot (10-4)!} \cdot \frac{6!}{3! \cdot (6-3)!} \cdot \frac{3!}{3! \cdot (3-3)!} = \frac{10!}{4! \cdot 3! \cdot 3!} = 4200$$

Aufgabe 11.2.a – Anzahl beliebiger Schedules

- Im Allgemeinen Fall:

$$\frac{(i_1 + i_2 + \cdots + i_n)!}{i_1! \cdot i_2! \cdot \cdots \cdot i_n!}$$

Aufgabe 11.2.b – Anzahl serieller Schedules

Anzahl der Permutationen der Transaktionen

-> Jede Transaktion kann an einer beliebigen stelle stehen

Im Beispiel:

$$3! = 6$$

Im allgemeinen:

$$n!$$

Aufgabe 11.2.c – Anzahl serialisierbarer Schedules

Kann man im allgemeinen nicht genau bestimmen, da die Anzahl von den Abhängigkeiten bestimmt wird.

Liegt aber zwischen der Anzahl beliebiger Schedules und der Anzahl der seriellen Schedules

Im Beispiel:

$$6 \leq Anz.\,serialisierbarer\ Schedules \leq 4200$$

Im allgemeinen:

$$n! \leq Anz.\,serialisierbarer\ Schedules \leq \frac{(i_1+i_2+\cdots+i_n)!}{i_1!\cdot i_2!\cdot\cdots\cdot i_n!}$$

Aufgabe 11.3 – Serialisierbarkeit von Schedules

Geben Sie für die Beispiele jeweils den vollständigen Abhängigkeitsgraphen, sowie ggf. einen äquivalenten seriellen Schedule an bzw. begründen Sie kurz warum dieser nicht existiert.

Aufgabe 11.3.a – Abhängigkeiten

$$S_1 = \left(w_1(x), w_2(y), w_3(x), r_1(x), r_2(z), w_4(y), r_4(z), w_4(x), r_3(y), r_1(z)\right)$$

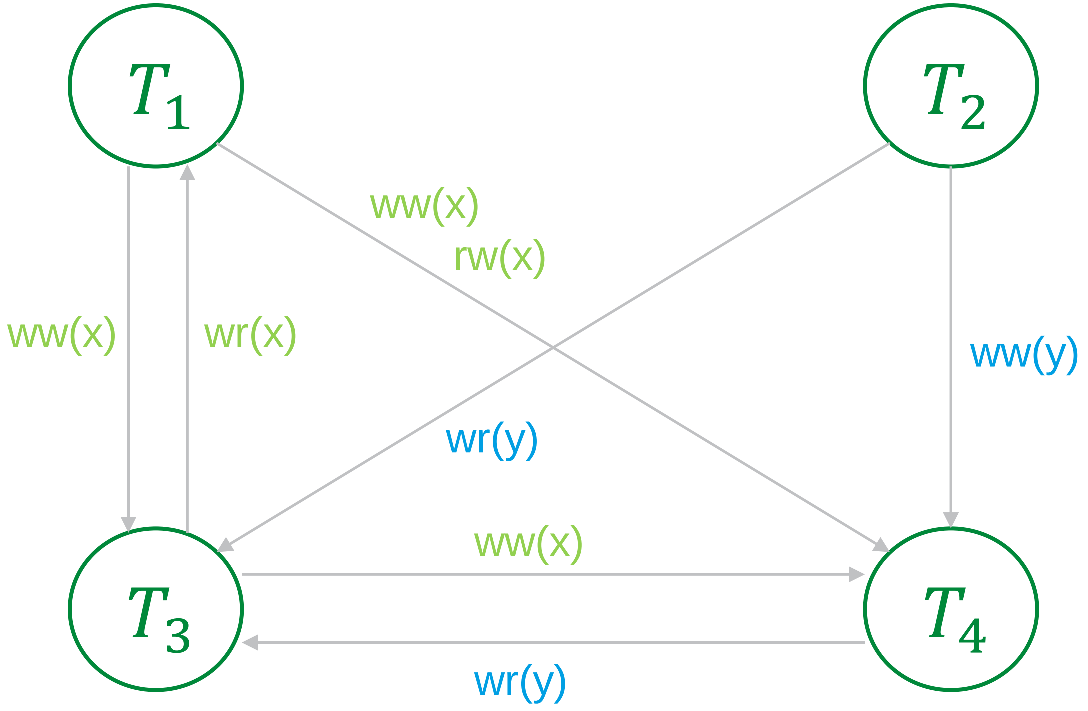
Abhängigkeiten:

- Auf Objekt x: $ww_{1,3}(x), ww_{1,4}(x), wr_{3,1}(x), ww_{3,4}(x), rw_{1,4}(x)$
- Auf Objekt y: $ww_{2,4}(y), wr_{2,3}(y), wr_{4,3}(y)$
- Auf Objekt z: wird nur gelesen -> keine Abhängigkeiten

Aufgabe 11.3.a - Serialisierungsgraph

Abhängigkeiten:

$ww_{1,3}(x), ww_{1,4}(x), wr_{3,1}(x), ww_{3,4}(x), rw_{1,4}(x), ww_{2,4}(y), wr_{2,3}(y), wr_{4,3}(y)$



-> nicht serialisierbar, da der Graph mehrere Zyklen enthält z.B. T1 -> T3

Aufgabe 11.3.b – Abhängigkeiten

$$S = \left(\mathrm{w}_1(\mathrm{x}), \mathrm{r}_4(\mathrm{x}), \mathrm{r}_1(\mathrm{y}), \mathrm{r}_1(\mathrm{z}), \mathrm{w}_1(\mathrm{z}), \mathrm{w}_3(\mathrm{x}), \mathrm{r}_4(\mathrm{z}), \mathrm{w}_3(\mathrm{y}), \mathrm{w}_2(\mathrm{y}), \mathrm{w}_2(\mathrm{z})\right)$$

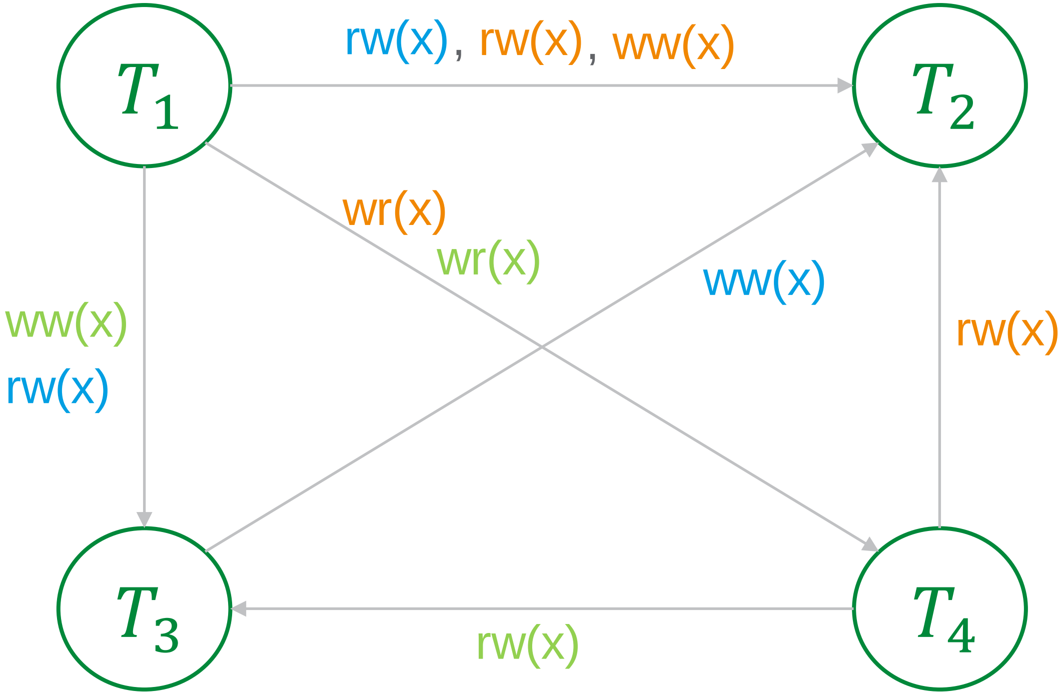
Abhängigkeiten

- Auf Objekt x: $wr_{1,4}(x), ww_{1,3}(x), rw_{4,3}(x)$
- Auf Objekt y: $rw_{1,3}(y), rw_{1,2}(y), ww_{3,2}(y)$
- Auf Objekt z: $rw_{1,2}(z), wr_{1,4}(z), ww_{1,2}(z), rw_{4,2}(z)$

Aufgabe 11.3.b – Serialisierungsgraph

Abhängigkeiten:

$$wr_{1,4}(x), ww_{1,3}(x), rw_{4,3}(x), rw_{1,3}(y), rw_{1,2}(y), ww_{3,2}(y),$$
$$rw_{1,2}(z), wr_{1,4}(z), ww_{1,2}(z), rw_{4,2}(z)$$



Graph ist Zyklenfrei -> serialisierbar

Konfliktäquivalenter serieller Schedule: (T_1, T_4, T_3, T_2)

Aufgabe 11.4.a – Anomalien

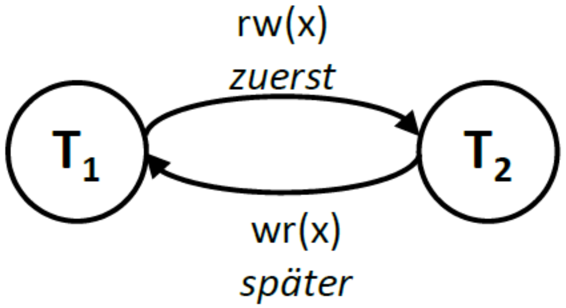
$$S = \left(r_1(x), r_2(y), w_2(x), r_1(z), r_1(x), w_2(y), w_1(z)\right)$$

Für Anomalie sind immer 3 Aktionen von 2 Transaktionen auf ein Objekt nötig

In diesem Fall ist das nur für x der Fall

Muster: $r_1(x), w_2(x), r_1(x)$

$\Rightarrow$ Non-Repeatable Read

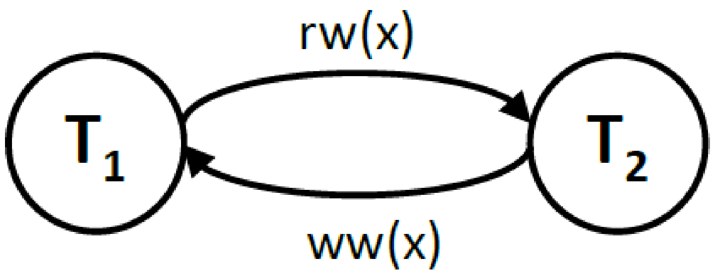


Aufgabe 11.4.b – Anomalien

$$S = (r_2(y), r_1(x), w_2(x), w_2(y), w_1(x))$$

Muster: $r_1(x), w_2(x), w_1(x)$

=> Lost Update bezüglich x

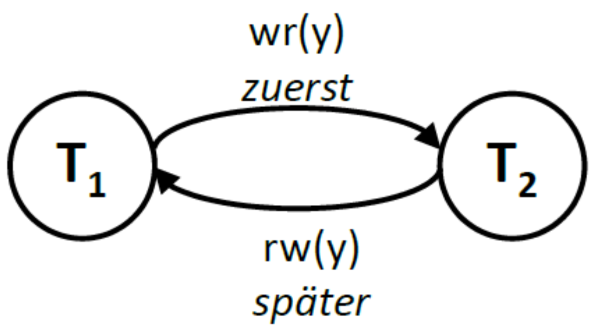


Aufgabe 11.4.c - Anomalien

$$S = \left(r_1(x), r_2(z), w_1(y), r_2(y), w_1(x), w_2(z), w_1(y)\right)$$

Muster: $w_1(y), r_2(y), w_1(y)$

=> Dirty Read bezüglich y



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Finn Kapitza – Finn.Kapitza@campus.lmu.de