

# Datenbanken und Informationssysteme

## Serialisierbarkeit

Burkhardt Renz

Fachbereich MNI  
TH Mittelhessen

Sommersemester 2021

# Übersicht

- Serialisierbarkeit
- 2-Phasen-Sperrprotokoll (2PL)
- Verklemmungen

# Modell einer Datenbank

- Datenobjekte  $x, y, z \dots$
- Aktionen
  - $r(x)$  – lese Datenobjekt  $x$  (read)
  - $w(x)$  – schreibe Datenobjekt  $x$  (write)
  - $c$  – bestätige Transaktion (commit)
  - $a$  – breche Transaktion ab (abort)
- Transaktion ist Folge solcher Aktionen:  
 $r(x); w(x); r(y); w(z); c$

## Aktionen mehrerer Transaktionen

- $T_1 \triangleq r_1(x); w_1(x); r_1(y); w_1(y)$
- $T_2 \triangleq r_2(x); w_2(x)$

Der Index gibt an, welche Transaktion die Aktion durchführt.

# Verschränkung von Transaktionen

Drei Abläufe  $S_1$ ,  $S_2$  und  $S_3$  mit den beiden Transaktionen:

- $S_1 \cong r_1(x); w_1(x); r_1(y); w_1(y); r_2(x); w_2(x)$
- $S_2 \cong r_1(x); r_2(x); w_1(x); r_1(y); w_2(x); w_1(y)$
- $S_3 \cong r_1(x); w_1(x); r_2(x); w_2(x); r_1(y); w_1(y)$

# Serielle Abläufe

## Definition

Ein Ablauf heißt **seriell**, wenn alle Schritte einer Transaktion vollständig ausgeführt werden, ehe die der nächsten Transaktion beginnen.

## Beispiel

Der Ablauf  $S_1$  ist seriell, die beiden Transaktionen folgen zeitlich aufeinander.

# Konflikt zwischen Aktionen

## Definition

Zwei Aktionen in einem Ablauf **stehen in Konflikt**, wenn gilt

- 1 sie gehören zu unterschiedlichen Transaktionen
- 2 sie greifen auf dasselbe Datenobjekt zu
- 3 mindestens einer der Aktionen ist **write**

## Beispiel

- $r_1(x)$  und  $w_2(x)$  stehen in Konflikt
- $r_1(x)$  und  $r_2(x)$  stehen nicht in Konflikt (beide lesend)
- $w_1(y)$  und  $w_2(x)$  stehen nicht in Konflikt (unterschiedliche Datenobjekte)

# Konfliktäquivalenz

## Definition

Zwei Abläufe heißen **konfliktäquivalent**, wenn die Reihenfolge von allen Paaren von in Konflikt stehenden Aktionen in beiden Abläufen gleich ist.

## Beispiel

- $S_1$  und  $S_2$  sind **nicht** konfliktäquivalent
- $S_1$  und  $S_3$  sind konfliktäquivalent



# Konfliktserialisierbarkeit

## Definition

Ein Ablauf heißt **konfliktserialisierbar**, wenn er zu einem seriellen Ablauf konfliktäquivalent ist.

## Beispiel

- $S_1$  ist seriell, also konfliktserialisierbar
- $S_2$  ist **nicht** serialisierbar
- $S_3$  ist serialisierbar, weil konfliktäquivalent zu  $S_1$

# Präzedenzgraph

## Fragestellung

Kriterium für Serialisierbarkeit

## Definition

Der **Präzedenzgraph** zu einem Ablauf  $S$  ist ein gerichteter Graph  $G_S$  mit

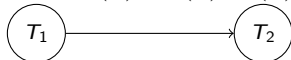
- ① den Knoten  $T_1, T_2, \dots$  für jede Transaktion  $T_i$  in  $S$
- ② den Kanten  $T_i \rightarrow T_j$  falls  $T_i$  und  $T_j$  konfligierende Aktionen haben, bei denen die Aktion in  $T_i$  **vor** der in  $T_j$  in  $S$  vorkommt.

# Beispiele für den Präzedenzgraphen

## Beispiel

Präzedenzgraphen zu unseren Beispielabläufen:

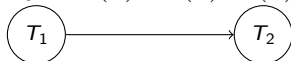
- $S_1 \cong r_1(x); w_1(x); r_1(y); w_1(y); r_2(x); w_2(x)$



- $S_2 \cong r_1(x); r_2(x); w_1(x); r_1(y); w_2(x); w_1(y)$



- $S_3 \cong r_1(x); w_1(x); r_2(x); w_2(x); r_1(y); w_1(y)$



# Kriterium für Serialisierbarkeit

## Satz (Kriterium für Serialisierbarkeit)

*Ein Ablauf  $S$  ist genau dann (konflikt-)serialisierbar, wenn sein Präzedenzgraph  $G_S$  keinen Zyklus hat.*

### Beweis.

⇒) Widerspruchsbeweis

⇐) Induktion über die Zahl der Transaktionen



# Übersicht

- Serialisierbarkeit
- 2-Phasen-Sperrprotokoll (2PL)
- Verklemmungen

# Modell binärer Sperren

Wir verwenden folgendes Modell für Sperren auf Datenobjekten:

## Binäre Sperren

- $l_i(x)$  – Transaktion  $T_i$  erwirkt eine Sperre auf Datenobjekt  $x$  (**lock**)
- $u_i(x)$  – Transaktion  $T_i$  gibt seine Sperre auf dem Datenobjekt  $x$  frei (**unlock**)

# Protokoll binärer Sperren

## Verhalten der Transaktionen

- 1 Eine Transaktion muss vor Lese- oder Schreibaktion auf ein Datenobjekt  $x$  eine Sperre auf  $x$  anfordern.
- 2 Eine Transaktion muss eine Sperre auf  $x$  freigeben, wenn sie keine Aktionen mit  $x$  mehr durchführen will.

## Verhalten des Systems

- 1 Die Anforderung  $I_i(x)$  von  $T_i$  wird nur ausgeführt, wenn keine andere Transaktion eine Sperre auf  $x$  hält, andernfalls muss  $T_i$  **warten**.
- 2 Wird eine Freigabe eines Datenobjekts ausgeführt, werden alle darauf wartenden Transaktionen wieder angestoßen.

# Sperrern und Serialisierbarkeit

## Bemerkung

Das Befolgen dieses Protokolls garantiert Serialisierbarkeit **nicht**.

## Beispiel

$l_1(x); w_1(x); u_1(x); l_2(y); w_2(y); u_2(y);$

$l_1(y); w_1(y); u_1(y); l_2(x); w_2(x); u_2(x);$



## 2-Phasen-Lock-Protokoll

### Definition

Eine Transaktion folgt dem **2-Phasen-Sperrprotokoll (2PL)**, wenn *alle* Sperraktionen in der Transaktion *vor* der ersten Freigabeaktion ausgeführt werden.

Die erste Phase, in der Sperren angefordert werden, nennt man die **Wachstumsphase**.

Die zweite Phase, in der nur noch Freigabe von Sperren erfolgen, nennt man die **Schrumpfungsphase**.

# Striktes 2PL

## Definition

Eine Transaktion folgt dem **strikten** 2PL, wenn sie dem 2PL folgt und alle Sperren en bloc am Ende der Transaktion freigibt.

## Bemerkung

Ein Ablauf nach dem 2PL ist äquivalent zu einem Ablauf nach dem strikten 2PL.

Begründung

## 2PL und Serialisierbarkeit

### Satz

*Ein Ablauf, in dem alle Transaktionen dem 2PL folgen, ist serialisierbar.*

### Beweis.

Induktion über die Zahl der Transaktionen



### Korollar

Ein Ablauf nach dem 2PL ist äquivalent zu einem seriellen Ablauf in der Reihenfolge der Transaktionen wie sie beginnen, ihre Sperren freizugeben.

## 2PL und Serialisierbarkeit — Diskussion

### Beispiel

$l_1(x); l_1(y); w_1(x); u_1(x); l_2(x); w_2(x);$   
 $w_1(y); u_1(y); l_2(y); w_2(y); u_2(x); u_2(y)$

Nun bricht  $T_1$  die Transaktion ab und  $T_2$  bestätigt sie  
Was passiert?

### Diskussion

- 2PL garantiert Konflikt-Serialisierbarkeit
- aber „Dirty Read“ ist möglich
- aber 2PL genügt nicht für korrektes Zurücksetzen von abgebrochenen Transaktionen
- *Striktes* 2PL hilft!

# Übersicht

- Serialisierbarkeit
- 2-Phasen-Sperrprotokoll (2PL)
- Verklemmungen

## 2PL und Verklemmungen

### Bemerkung

2PL verhindert Verklemmungen **nicht**

### Beispiel

$T_1 \triangleq l_1(x); r_1(x); w_1(x); l_1(y); u_1(x); r_1(y); w_1(y); u_1(y);$

$T_2 \triangleq l_2(y); r_2(y); w_2(y); l_2(x); u_2(y); r_2(x); w_2(x); u_2(x);$

Zeitlicher Ablauf:

$T_1$	$l_1(x); r_1(x)$	$w_1(x)$	$l_1(y) \dots$
$T_2$	$l_2(y); r_2(y)$	$w_2(y)$	$l_2(x) \dots$

# Wartegraph

## Thema und Modell

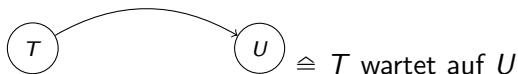
Thema: Erkennen und Auflösen von Verklemmungen (Deadlocks)

Rahmen: Unser einfaches Modell mit binären Sperren

## Definition

Der **Wartegraph** ist ein gerichteter Graph mit

- 1 die Knoten sind die Transaktionen
- 2 Zwei Knoten  $T$  und  $U$  sind durch eine gerichtete Kante verbunden, wenn  $T$  darauf wartet, dass  $U$  ein Datenobjekt freigibt.



# Beispiel Wartegraph

## Beispiel

$T_1, T_2, T_3$  möchten folgende Abläufe machen:

- $T_1 : l_1(x); r_1(x); l_1(y); w_1(y); u_1(x); u_1(y);$
- $T_2 : l_2(z); r_2(z); l_2(x); w_2(x); u_2(z); u_2(x);$
- $T_3 : l_3(y); r_3(y); l_3(z); w_3(z); u_3(y); u_3(z);$

Es entsteht z.B. folgender Ablauf in zeitlicher Reihenfolge:

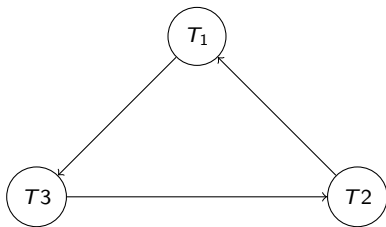
$l_1(x); r_1(x); l_2(z); r_2(z); l_3(y); r_3(y); l_2(x); l_3(z); l_1(y);$

Ergebnis:

- $T_2$  wartet auf  $T_1$
- $T_3$  wartet auf  $T_2$
- $T_1$  wartet auf  $T_3$



## Wartegraph zum Beispiel



# Wartegraph und Verklemmungen

## Satz

*Es liegt genau dann eine Verklemmung vor, wenn der Wartegraph einen Zyklus hat.*

## Bemerkung

Der Wartegraph kann auf zwei Arten verwendet werden:

- 1 *Deadlock-Erkennung*
- 2 *Deadlock-Vermeidung*

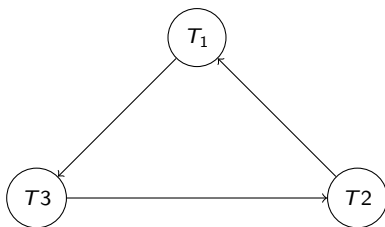
# Adjazenzmatrix des Wartegraphen

Man repräsentiert den Wartegraphen gerne durch seine Adjazenzmatrix:

Die Zeilen und Spalten der Matrix repräsentieren die Transaktionen  $T_1, T_2, T_3, \dots, T_n$  und für ein Element  $w_{ij}$  der Matrix gilt:

$$w_{ij} = \begin{cases} 1 & \text{falls } T_i \rightarrow T_j \\ 0 & \text{sonst} \end{cases}$$

## Beispiel für die Adjazenzmatrix



$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

# Algorithmus

- 1 Entferne alle Transaktionen, die an keinem Zyklus beteiligt sind, d.h. diejenigen, bei denen
  - (a) die Zeile nur 0 enthält, oder
  - (b) die Spalte nur 0 enthält.
- 2 Sind jetzt noch Transaktionen übrig, müssen sie an einem Zyklus beteiligt sind.  
Wähle ein Opfer und breche die Transaktion ab.  
Weiter mit Schritt 1.