





© N. Ritter

# Transaktionale Informationssysteme

## 12. Verteilte Synchronisation

Norbert Ritter  
Datenbanken und Informationssysteme  
vsis-www.informatik.uni-hamburg.de






© N. Ritter

## Homogene Föderationen (1)

- Homogene Föderation
  - n *Sites*, auf die die Daten verteilt sind
  - $D = \bigcup_{i=1}^n D_i, 1 \leq i \leq n$
  - keine Replikation
  - ausschließlich globale TA
- Definition **Globale Historie**
  - Die betrachtete Föderation bestehe aus n *Sites*.  
Es sei  $T = \{t_1, \dots, t_m\}$  eine Menge von (globalen) TA.  
Es seien  $s_1, \dots, s_n$  lokale Historien.
  - Eine globale Historie für T und  $s_1, \dots, s_n$  ist eine Historie s für T, wobei gilt:
 
$$\Pi_i(s) = s_i \text{ für alle } i, 1 \leq i \leq n.$$

TAIS – WS0506 – Kapitel 10: Verteilte Synchronisation

2



Homogen

Deadlocks

Heterogen



© N. Ritter

## Homogene Föderationen (2)

- Subtransaktion
  - Projektion einer (globalen) Transaktion auf *Site*  $i$
- Beispiel
  - betrachte Föderation über 2 *Sites*, wobei  $D1 = \{x\}$  und  $D2 = \{y\}$ .
  - dann sind  $s_1 = r_1(x) w_2(x)$  und  $s_2 = w_1(y) r_2(y)$  lokale Schedules, und
  - $s = r_1(x) w_1(y) w_2(x) c_1 r_2(y) c_2$  eine globale Historie.
  - $\Pi_1(s) = s_1$  und  $\Pi_2(s) = s_2$  (bis auf Commit-Operationen).
  - alternative Schreibweise für globale Historien:
 

Server 1:	$r_1(x)$	$w_2(x)$	
Server 2:	$w_1(y)$		$r_2(y)$



Homogen

Deadlocks

Heterogen



© N. Ritter

## Homogene Föderationen (3)

- Definition **Konfliktserialisierbarkeit**
  - Eine globale (lokale) Historie  $s$  ist global (lokal) *konfliktserialisierbar*, wenn eine serielle Historie über den globalen (lokalen) (Sub-) Transaktionen existiert, die konflikt-äquivalent zu  $s$  ist.
- Beispiel
  - betrachte folgende globale Historie  $s$ 

Server 1:	$r_1(x)$	$w_2(x)$	
Server 2:	$r_2(y)$		$w_1(y)$
  - Scheduler auf *Site* 1:  $t_1 < t_2$
  - Scheduler auf *Site* 2:  $t_2 < t_1$
  - Konfliktgraph der globalen Historie würde demnach einen Zykel enthalten, so dass  $s$  nicht konflikt-serialisierbar ist



Homogen

Deadlocks

Heterogen



© N. Ritter

## Homogene Föderationen (4)

### ■ Theorem

- Sei  $s$  globale Historie mit lokalen Historien  $s_1, \dots, s_n$  über einer Menge  $T$  von Transaktionen, so dass jedes  $s_i$ ,  $1 \leq i \leq n$ , konflikt-serialisierbar ist. Dann gilt:
- $s$  ist genau dann global konflikt-serialisierbar, wenn es eine totale Ordnung „ $<$ “ auf  $T$  gibt, die mit den lokalen Serialisierungsreihenfolgen der Transaktionen konsistent ist, d.h.,
- $(\forall t, t' \in T, t \neq t') t < t' \Rightarrow (\forall s_i, 1 \leq i \leq n, t, t' \in \text{trans}(s_i)) (s_i' \text{ seriell, } s_i \approx_c s_i') t <_{s_i'} t'$



Homogen

Deadlocks

Heterogen



© N. Ritter

## Homogene Föderationen (5)

### ■ Nutzung von 2PL

- Problem unter der Annahme der lokalen Nutzung von 2PL
  - global entscheiden, wann eine Sperre freigegeben werden kann
- Lösung 1: *Primary Site 2PL*
  - Nachteil: *Primary Site* wird schnell zum Flaschenhals
- Lösung 2: *Distributed 2PL (D2PL)*
  - Halten des Zustands aller lokalen Schedules auf allen Sites
    - Nachteil: hoher Kommunikationsaufwand
  - lokaler Server fragt vor Eintritt in die Unlock-Phase bei allen anderen nach
    - Nachteil: Aufwand immer noch (zu) hoch

## Homogene Föderationen (6)

### ■ Nutzung von 2PL (Forts.)

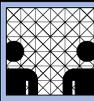
- Anmerkung

- Wenn alle lokalen Server SS2PL anwenden, dann ist die resultierende globale Historie nicht nur konflikt-serialisierbar sondern strikt.

### ■ Ebenfalls verfügbar

(siehe Weikum, Vossen: Transactional Information Systems, S. 680 ff.)

- Distributed TO
- Distributed SGT
- Distributed optimistic protocols



Homogen

Deadlocks

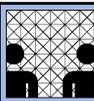
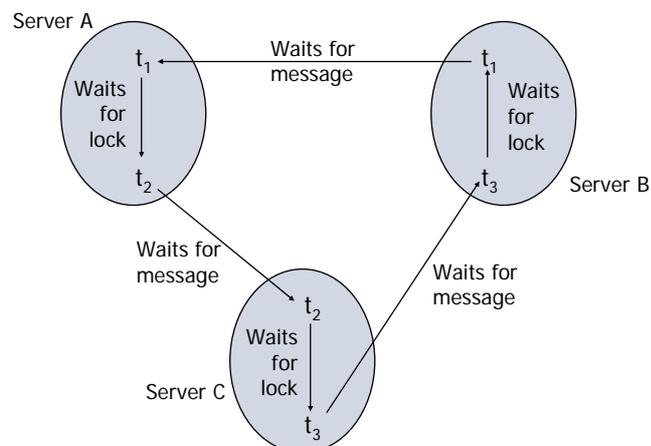
Heterogen



© N. Ritter

## Verteilte Deadlock-Erkennung (1)

### ■ Beispiel eines verteilten Deadlocks



Homogen

Deadlocks

Heterogen



© N. Ritter



Homogen

Deadlocks

Heterogen



© N. Ritter

## Verteilte Deadlock-Erkennung (2)

- Lösung 1: *Zentralisierte Erkennung*
  - Einsatz einer Art *zentralisierter Monitor*
    - Sammeln und Analysieren von ‚Warte-Informationen‘ der lokalen Server
    - Ermitteln eines ‚Opfers‘ nach Erkennung eines Deadlocks durch geeignete Kommunikation mit den lokalen Servern (insbes. zur Berücksichtigung der Rollback-Kosten)
  - Nachteile
    - Flaschenhals
    - (Kommunikations-)Aufwand
  - Eignung
    - nur im Falle sehr schneller und hoch verfügbarer Kommunikationsverbindungen
    - nicht bei Kommunikation über das Internet

TAIS – WS0506 – Kapitel 10: *Verteilte Synchronisation*

9



Homogen

Deadlocks

Heterogen



© N. Ritter

## Verteilte Deadlock-Erkennung (3)

- Lösung 1: *Zentralisierte Erkennung*
  - Weiteres Problem: *Falsche Deadlocks*
    - sofort, nachdem die letzte, den Zykel schließende Kante durch den Monitor vermerkt wird, wird eine der beteiligten TAs durch (lokal entschiedenes) Abort abgebrochen
    - Monitor bemerkt dies nicht und bricht eine zweite TA ab
    - tritt nicht auf im Falle der Nutzung von 2PL durch alle lokalen Scheduler und des Nicht-Auftretens von ‚spontanen Aborts‘ (TA-Selbstmord)
- Timeouts können genutzt werden

TAIS – WS0506 – Kapitel 10: *Verteilte Synchronisation*

10



Homogen

Deadlocks

Heterogen



© N. Ritter

## Verteilte Deadlock-Erkennung (4)

- Lösung 2: *Dezentralisierte Ansätze*
  - *Edge chasing*
  - *Path pushing*
- *Edge chasing*
  - eine TA, die durch Warte-Beziehung blockiert wird, schickt *Probe*-Nachricht mit eigener TA-ID zur blockierenden TA
  - jede TA, die *Probe*-Nachricht bekommt, schickt diese weiter an alle TAs, die sie selbst blockieren
  - erhält eine TA eine Nachricht mit eigener TA-ID, ist Deadlock erkannt
  - Lösung kann der eigene Abbruch sein

TAIS – WS0506 – Kapitel 10: *Verteilte Synchronisation*

11



Homogen

Deadlocks

Heterogen



© N. Ritter

## Verteilte Deadlock-Erkennung (5)

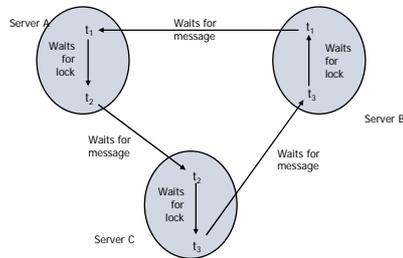
- *Path pushing*
  - Idee: es zirkulieren ganze Pfade statt einzelner TA-Ids
  - Algorithmus
    1. Jeder Server, der einen *waits-for path* von  $t_i$  zu  $t_j$  hat, wobei  $t_i$  eine eingehende und  $t_j$  eine ausgehende *waits-for message* hat, sendet diesen Pfad entlang der ausgehenden Kante, vorausgesetzt der Identifier von  $t_i$  ist kleiner als der von  $t_j$ .
    2. Nach Erhalt eines Pfads konkateniert der Server diesen mit den lokalen Pfaden und leitet das Ergebnis selbst wieder weiter. Falls ein Zykel zwischen  $n$  Server existiert, erkennt mindestens einer der Server diesen in höchstens  $n$  Schritten.

TAIS – WS0506 – Kapitel 10: *Verteilte Synchronisation*

12

## Verteilte Deadlock-Erkennung (6)

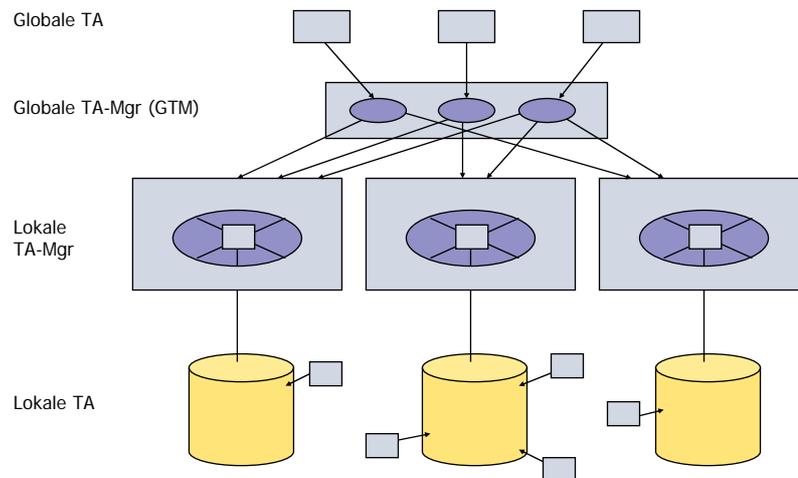
- *Path pushing (Forts.)*
  - Beispiel auf Folie 8



- Server C kennt  $t_3 \rightarrow t_1$  und erkennt damit den globalen Deadlock

## Heterogene Föderationen (1)

- Illustration

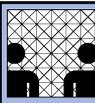


## Heterogene Föderationen (2)

### ■ Historien in heterogenen Föderationen

#### • Beispiel

- $D_1 = \{a, b\}$ ,  $D_2 = \{c, d, e\}$
- $D = \{a, b, c, d, e\}$
- lokale TAs:  $t_1 = r(a) w(b)$ ,  $t_2 = w(d) r(e)$
- globale TAs:  $t_3 = w(a) r(d)$ ,  $t_4 = w(b) r(c) w(e)$
- lokale Historien:  
 $s_1 = r_1(a) w_3(a) c_3 w_1(b) c_1 w_4(b) c_4$   
 $s_2 = r_4(c) w_2(d) r_3(d) c_3 r_2(e) c_2 w_4(e) c_4$



Homogen

Deadlocks

Heterogen



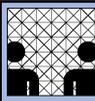
© N. Ritter

## Heterogene Föderationen (3)

### ■ Historien in heterogenen Föderationen (Forts.)

#### • Definition **Globale Historie** (revisited)

- die betrachtete heterogene Föderation bestehe aus  $n$  *Sites*;
- $T_1, \dots, T_n$  seien lokale TA auf den *Sites*  $1, \dots, n$
- $T$  sei Menge globaler TA
- $s_1, \dots, s_n$  seien lokale Historien mit  
 $T_i \subseteq \text{trans}(s_i)$  und  $T \cap \text{trans}(s_i) \neq \emptyset$  für  $1 \leq i \leq n$ .
- eine (*heterogene*) *globale Historie* (für  $s_1, \dots, s_n$ ) ist eine Historie  $s$  für  $\bigcup_{i=1}^n T_i \cup T$ , so dass ihre lokale Projektion jeweils gleich ist zu den lokalen Historien jeder Site, d.h.,  
 $\Pi_i(s) = s_i$  für alle  $i$ ,  $1 \leq i \leq n$ .



Homogen

Deadlocks

Heterogen



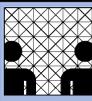
© N. Ritter

## Heterogene Föderationen (4)

### ■ Historien in heterogenen Föderationen (Forts.)

#### • Beispiel

- $D_1 = \{a\}, D_2 = \{b, c\}$
- globale TAs:  $t_1 = r(a) w(b), t_2 = w(a) r(c)$
- lokale TA:  $t_3 = r(b) w(c)$
- Annahme: GTM entscheidet,  $t_1$  zuerst auszuführen
- entstehende lokale Historien:  
Server 1:  $s_1 = r_1(a) \quad w_2(a)$   
Server 2:  $s_2 = \quad r_3(b) w_1(b) \quad r_2(c) \quad w_3(c)$
- beachte: beide globale TAs werden auf beiden Sites seriell ausgeführt
- globale Historie:  $s = r_1(a) r_3(b) w_1(b) c_1 w_2(a) r_2(c) c_2 w_3(c) c_3$
- offenbar  $s_1, s_2 \in \text{CSR}$ , aber  $s_1 \approx_c t_1 t_2$  während  $s_2 \approx_c t_2 t_3 t_1$
- daraus folgt, dass Konfliktgraph zu  $s$  zyklisch und die von GTM gewählte Ausführungsreihenfolge ist nicht akzeptierbar



Homogen

Deadlocks

Heterogen



© N. Ritter

TAIS – WS0506 – Kapitel 10: Verteilte Synchronisation

17

## Heterogene Föderationen (5)

### ■ Historien in heterogenen Föderationen (Forts.)

#### • Beispiel (Forts.)

- Ursachen
  - direkter Konflikt zwischen den globalen Transaktionen in  $s_1$
  - *indirekter Konflikt* in  $s_2$ , da
    - globale TA  $t_2$  in direktem Konflikt mit lokaler TA  $t_3$  und
    - lokale TA  $t_3$  in direktem Konflikt mit globaler TA  $t_1$
- indirekte Konflikte können auch auftreten, wenn es keine direkten Konflikte zwischen den globalen TA gibt  
(Beispiel: siehe Weikum, Vossen: Transactional Information Systems, S. 693)



Homogen

Deadlocks

Heterogen



© N. Ritter

TAIS – WS0506 – Kapitel 10: Verteilte Synchronisation

18

Homogen

Deadlocks

Heterogen

© N. Ritter

## Heterogene Föderationen (6)

### ■ Globale Serialisierbarkeit

#### • Definition *Direkte und Indirekte Konflikte*

- sei  $s_i$  lokale Historie und  $t$  und  $t'$  Transaktionen aus  $\text{trans}(s_i)$  mit  $t \neq t'$
- 1.  $t$  und  $t'$  stehen in direktem Konflikt in  $s_i$  wenn gilt  $(\exists p \in t) (\exists q \in t') (p, q) \in \text{conf}(s_i)$  (vgl. Kapitel 2)
- 2.  $t$  und  $t'$  stehen in indirektem Konflikt in  $s_i$ , wenn es eine Sequenz  $t_1, \dots, t_r$  von Transaktionen in  $\text{trans}(s_i)$  gibt, so dass  $t$  in  $s_i$  in direktem Konflikt mit  $t_1$ ,  $t_j$  in  $s_i$  in direktem Konflikt mit  $t_{j+1}$ ,  $1 \leq j \leq r-1$  und  $t_r$  in  $s_i$  in direktem Konflikt mit  $t'$
- 3.  $t$  und  $t'$  stehen in  $s_i$  in Konflikt, wenn sie in  $s_i$  in direktem oder indirektem Konflikt stehen

Homogen

Deadlocks

Heterogen

© N. Ritter

## Heterogene Föderationen (7)

### ■ Globale Serialisierbarkeit

#### • Definition *globaler Konfliktgraph*

- sei  $s$  globale Historie für die lokalen Historien  $s_1, \dots, s_n$ ;
- sei  $G(s_i)$  der Konfliktgraph von  $s_i$ ,  $1 \leq i \leq n$ , der durch Betrachtung direkter und indirekter Konflikte entsteht
- der *globale Konfliktgraph* von  $s$  ist definiert als die Vereinigung aller  $G(s_i)$ ,  $1 \leq i \leq n$ , d.h.

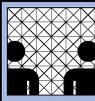
$$G(s) := \bigcup_{i=1}^n G(s_i)$$

#### • **(Multidatabase Serializability) Theorem**

- gegeben die lokalen Historien  $s_1, \dots, s_n$ , wobei jeder  $G(s_i)$ ,  $1 \leq i \leq n$ , azyklisch (d.h.,  $s_i \in \text{CSR}$ )
- weiter sei  $s$  die globale Historie für die  $s_i$ ,  $1 \leq i \leq n$
- dann gilt:  $s$  ist global konflikt-serialisierbar gdw  $G(s)$  azyklisch

## Heterogene Föderationen (8)

- Globale Serialisierbarkeit durch lokale Garantien
  - Ausnutzung von Commitment Ordering (vgl. Kapitel 2)
    - eine von mehreren Möglichkeiten  
(weitere siehe Weikum, Vossen: Transactional Information Systems, S. 698, ff.)
    - Theorem
      - sei  $s$  globale Historie für  $s_1, \dots, s_n$
      - falls  $s_i \in \text{COCSR}$ ,  $1 \leq i \leq n$ , und falls alle globale TA ihre Commits strikt sequentiell ausführen, dann ist  $s$  global serialisierbar



Homogen

Deadlocks

Heterogen



© N. Ritter

TAIS – WS0506 – Kapitel 10: Verteilte Synchronisation

21

## Heterogene Föderationen (9)

- Globale Serialisierbarkeit durch lokale Garantien
  - Ausnutzung von Commitment Ordering (vgl. Kapitel 2)
    - Beispiel
      - $s_1 = r_1(a) c_1 w_3(a) w_3(b) c_3 r_2(b) c_2$
      - $s_2 = w_4(c) r_1(c) r_2(d) r_4(e) c_1 c_2 [w_4(d) c_4]$
      - $t_1, t_2$  global;  $t_3, t_4$  lokal
      - beachte: Commit-Operationen sind in beiden Historien gleich angeordnet
      - angenommen  $s_2$  sei bis vor eckige Klammer abgearbeitet
      - beachte, dass die globalen TA nebenläufig auf *Site 2* laufen



Homogen

Deadlocks

Heterogen



© N. Ritter

TAIS – WS0506 – Kapitel 10: Verteilte Synchronisation

22



Homogen

Deadlocks

Heterogen



© N. Ritter

## Heterogene Föderationen (10)

- Globale Serialisierbarkeit durch lokale Garantien (Forts.)
  - Ausnutzung von Commitment Ordering (Forts.)
    - Beispiel (Forts.)
      - falls Server 2 COCSR garantiert, dann kann  $s_2$  nicht weitergeführt werden, da der indirekte Konflikt zwischen  $t_2$  und  $t_1$ , der dann durch  $t_4$  verursacht werden würde, es erforderlich machen würde, dass diese TA ihre Commit-Operationen in der durch diesen Konflikt vorgegebenen Reihenfolge ausführen müssten, was jedoch nicht mehr möglich ist
      - ein COCSR-Scheduler müsste also  $t_4$  abbrechen!
    - Beispiel eines Synchronisationsverfahrens, dass COCSR garantiert: *Ticket-Based CC*  
 (siehe Weikum, Vossen: Transactional Information Systems, S. 698, ff.)



Homogen

Deadlocks

Heterogen



© N. Ritter

## Zusammenfassung

- XOPEN/DTP (2PC) und lokales ACID garantieren im Allgemeinen nicht globale Serialisierbarkeit!