Kommunikation und Datenhaltung

4. Übung zur Datenhaltung

Anfrageoptimierung & Transaktionsverwaltung

Agenda

- **Informationen zu Lehrveranstaltungen am IPD im Wintersemester**
- Besprechung des vierten Übungsblatts
- Lehrevaluation

Praktikum „Verteilte Datenhaltung“

- Relationale Datenbanken / SQL
  - Erstellung von Datenbank-Schemata
  - Umsetzung komplexer Anfragen mit SQL
  - Implementierung mit Prozeduralem SQL (PL/SQL)
- Relationale Anfrageverarbeitung in Sensornetzen
  - Java-Programmierung von echten Sensorknoten (Sun SPOTs)
  - Entwurf/Implementierung eines eigenen Anfrageprozessors

- Wettbewerb: Wer hat den besten Anfrageprozessor für Sensornetze?

Praktikum „Verteilte Datenhaltung“

- Ablauf:
  - Teil 1:
    - Praktikum findet wöchentlich statt
    - Vorlesungen und Aufgaben in Themenblöcke gegliedert
  - Teil 2:
    - Gruppenarbeit
    - Entwurf, Implementierung und Evaluation des eigenen Anfrageprozessors
- Detailinformationen:
  - http://dbis.ipd.uni-karlsruhe.de/1339.php
  - **Bestes Praktikum WS 08/09**

Praktikum „Verteilte Datenhaltung“

- Teilnahmebedingungen
  - Bestandene K&D Klausur oder vergleichbare Kenntnisse
  - Paralleles Hören der Vorlesung „Verteilte Datenhaltung“
    ➔ optimale Prüfungsvorbereitung!
  - Anmeldung im Sekretariat IPD Böhm (Gebäude 50.34, Raum 367)
  - Grundlegende Kenntnisse in Objekt-Orientierter Programmierung
  - SQL soweit in K&D vorgestellt

Seminar „Kollaborative Datenschutzmechanismen“

- Problem
  - Durchdringung des Alltags mit intelligenten Geräten: RFID, Ubicomp, Sensornetze, Internet
  - Erosion der Privatsphäre
- Umfeld, Rahmenbedingungen
  - Rechtlich (Gesetze)
  - Technik (Anonymisierung, PETs, …)
  - Gesellschaft

Seminar „Kollaborative Datenschutzmechanismen“

- Kollaborative Datenschutzmechanismen
  - Für den Einzelnen schwierig, Verstöße festzustellen
  - Technikunterstützung wünschenswert
- Benutzer decken gemeinsam Verstöße auf
  - Beispielsweise rechtlichen Verstoß: Datenschutzerklärung lückenhaft
- http://dbis.ipd.uni-karlsruhe.de/1341.php
- Anmeldung im Sekretariat IPD Böhm, Liste liegt aus

Agenda

- Informationen zu Lehrveranstaltungen am IPD im Wintersemester
- **Besprechung des vierten Übungsblatts**
- Lehrevaluation

Aufgabe 1a: Anfrageoptimierung

```
SELECT albumname
FROM   Album, Language
WHERE  language = languageid
AND    languagename = 'German'
```

Aufgabe: Überführen Sie die SQL-Abfrage in einen relationsalgebraischen Ausdruck, der der in der Vorlesung vorgestellten Standardform genügt.

Grundmuster der SQL-Übersetzung

SELECT-Ausdruck

SELECT $A_1, A_2, \ldots, A_n$
FROM $R_1, R_2, \ldots, R_m$
WHERE B

wird überführt in:

$$\pi_{A_1, A_2, \ldots, A_n} (\sigma_B (R_1 \times R_2 \times \ldots \times R_m))$$

Aufgabe 1a: Anfrageoptimierung

```
SELECT albumname
FROM   Album, Language
WHERE  language = languageid
AND    languagename = 'German'
```

Aufgabe: Überführen Sie die SQL-Abfrage in einen relationsalgebraischen Ausdruck, der der in der Vorlesung vorgestellten Standardform genügt.

Lösung:

$$\Pi_{albumname}\left(\sigma_{\substack{language = languageid \\ \wedge\, languagename = 'German'}}(Album \times Language)\right)$$

Aufgabe 1a: Anfrageoptimierung

```
SELECT albumname
FROM   Album, Language
WHERE  language = languageid
AND    languagename = 'German'
```

Aufgabe: Überführen Sie die SQL-Abfrage in einen relationsalgebraischen Ausdruck, der der in der Vorlesung vorgestellten Standardform genügt.

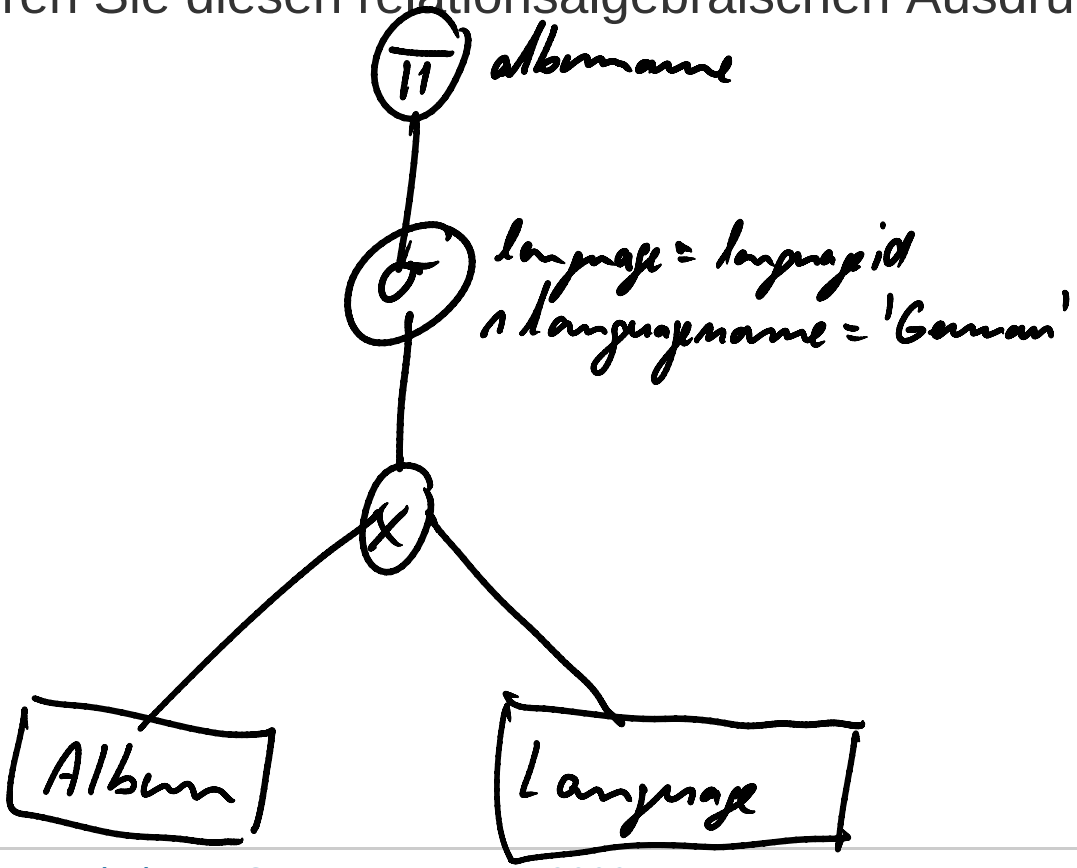
Lösung:

$$\pi_{\text{albumname}}(\sigma_{\text{language} = \text{languageid} \wedge \text{languagename} = \text{“German”}}(\text{Album} \times \text{Language}))$$

Aufgabe 1b: Anfrageoptimierung

$$\pi_{\text{albumname}}(\sigma_{\text{language} = \text{languageid} \wedge \text{languagename} = \text{“German”}}(\text{Album} \times \text{Language}))$$

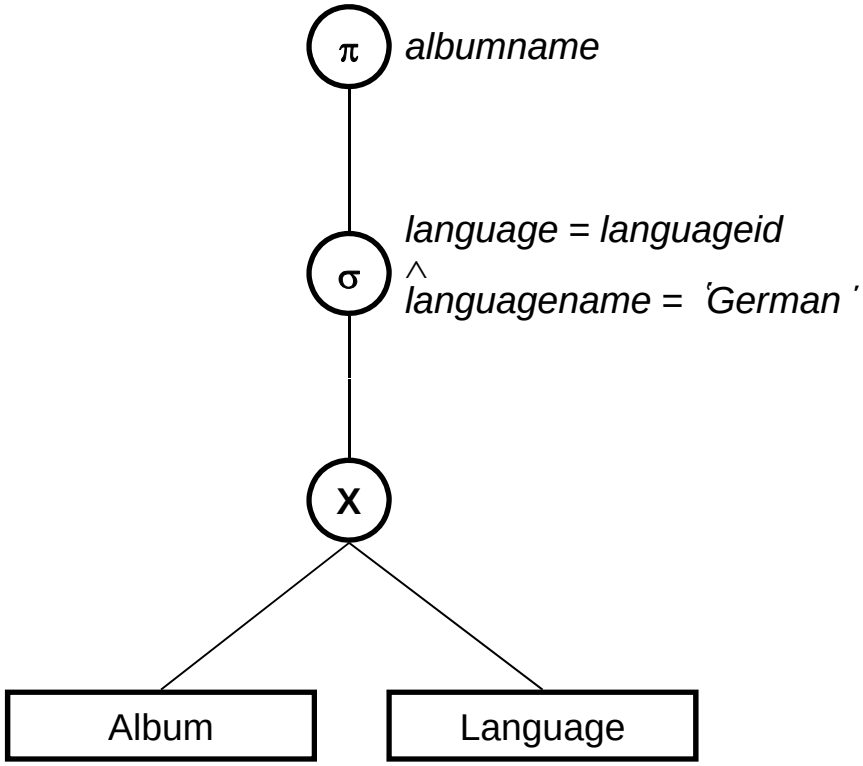
Aufgabe: Überführen Sie diesen relationsalgebraischen Ausdruck in Baumform.



Aufgabe 1b: Anfrageoptimierung

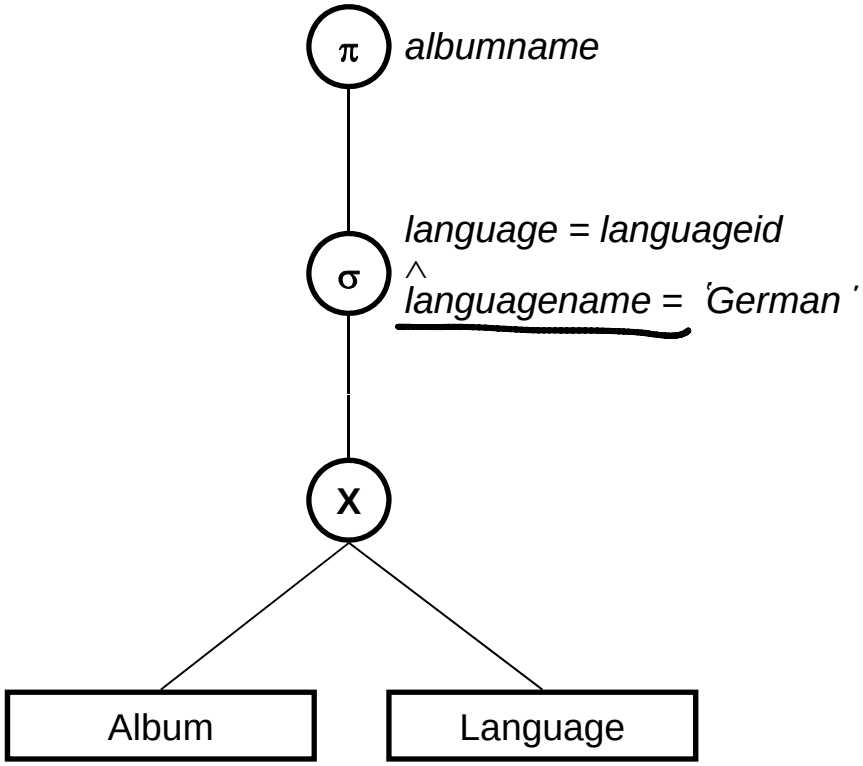
$$\pi_{\text{albumname}}(\sigma_{\text{language} = \text{languageid} \wedge \text{languagename} = \text{“German”}}(\text{Album} \times \text{Language}))$$

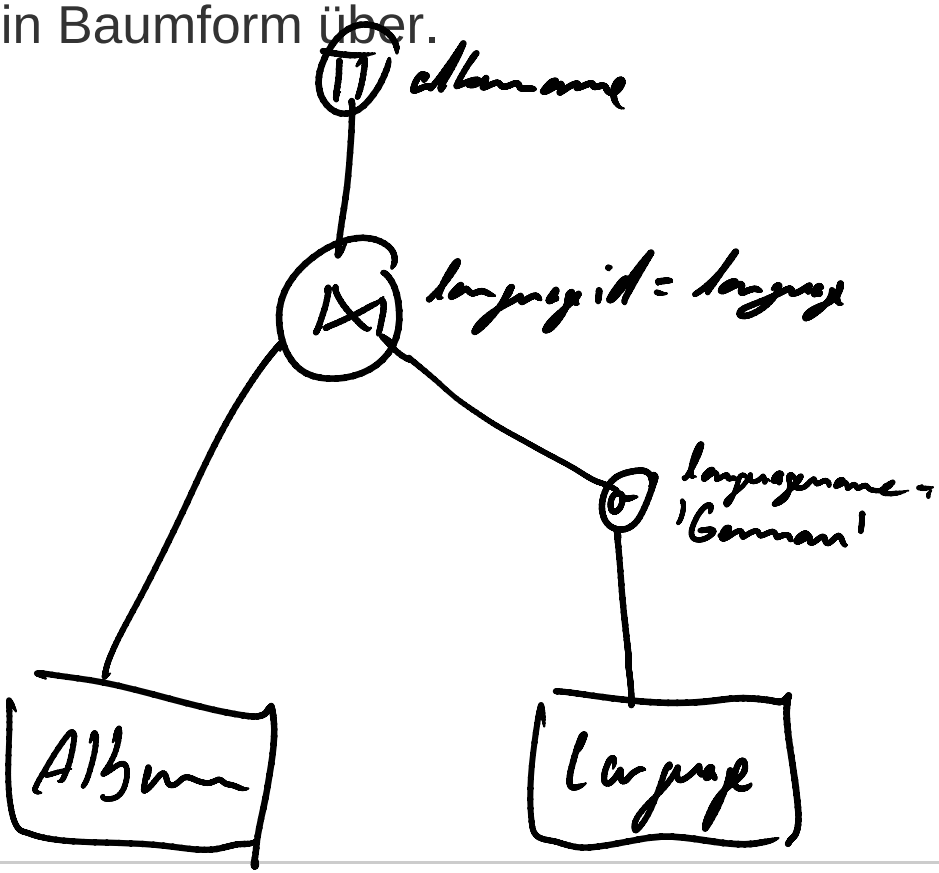
Aufgabe: Überführen Sie diesen relationsalgebraischen Ausdruck in Baumform.



Aufgabe 1c: Anfrageoptimierung

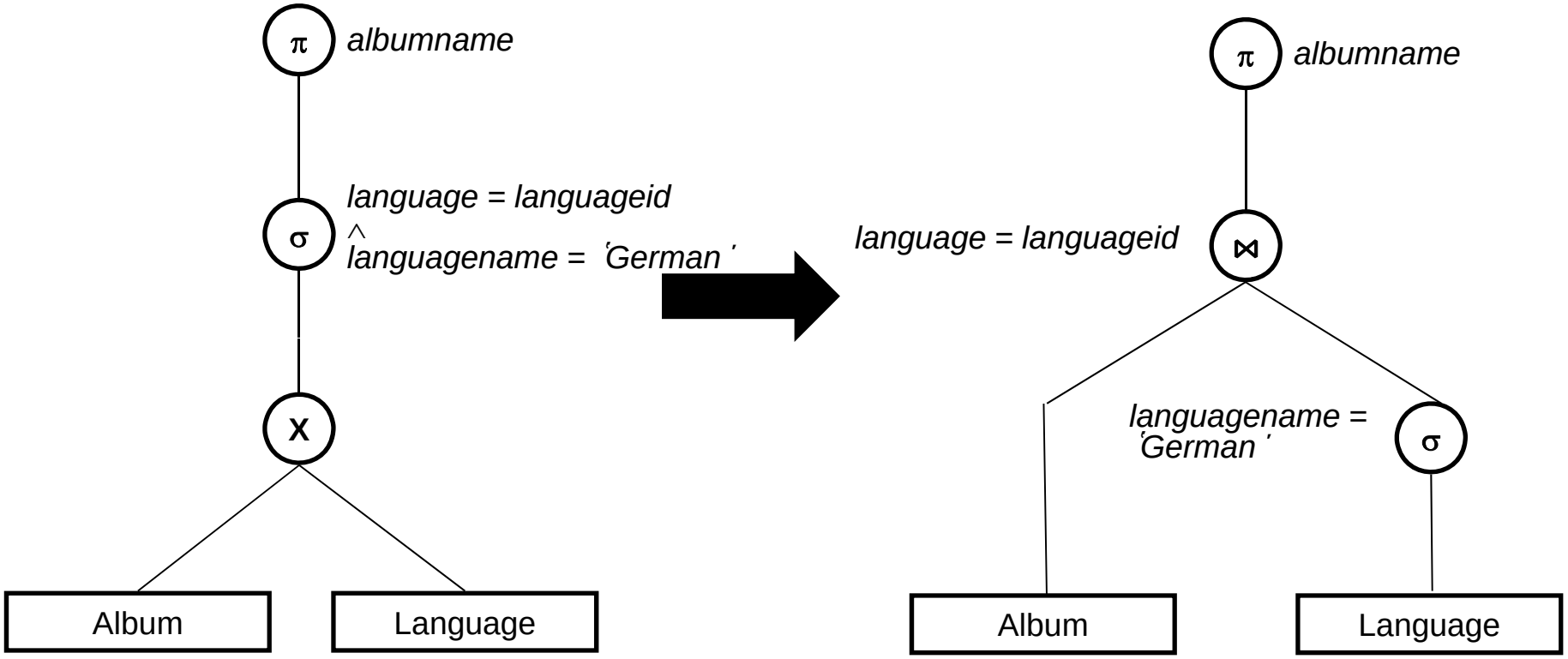
Aufgabe: Überlegen Sie sich geeignete algebraische Optimierungen für die Anfrage. Erstellen Sie einen relationsalgebraischen Ausdruck für die Optimierungen und führen Sie diesen in Baumform über.





Aufgabe 1c: Anfrageoptimierung

Aufgabe: Überlegen Sie sich geeignete algebraische Optimierungen für die Anfrage. Erstellen Sie einen relationsalgebraischen Ausdruck für die Optimierungen und führen Sie diesen in Baumform über.



Aufgabe 1c: Anfrageoptimierung

- Angewandte Regeln:
  - Führe Selektionen so früh wie möglich durch.
  - Fasse kartesische Produkte und Selektionen möglichst zu θ-Verbindungen zusammen.
- Vorteil:
  - Vorher: 476 (# Tupel in Relation „Language“) mal 664 167 (# Tupel in Relation „Album“) = 316 143 492 Vergleiche bei Selektion
  - Nach Optimierung nur 664 167 Vergleiche mal 1 (# Tupel in Relation „Language“ nach Selektion) beim Join!

Nebenläufigkeit & Transaktionen

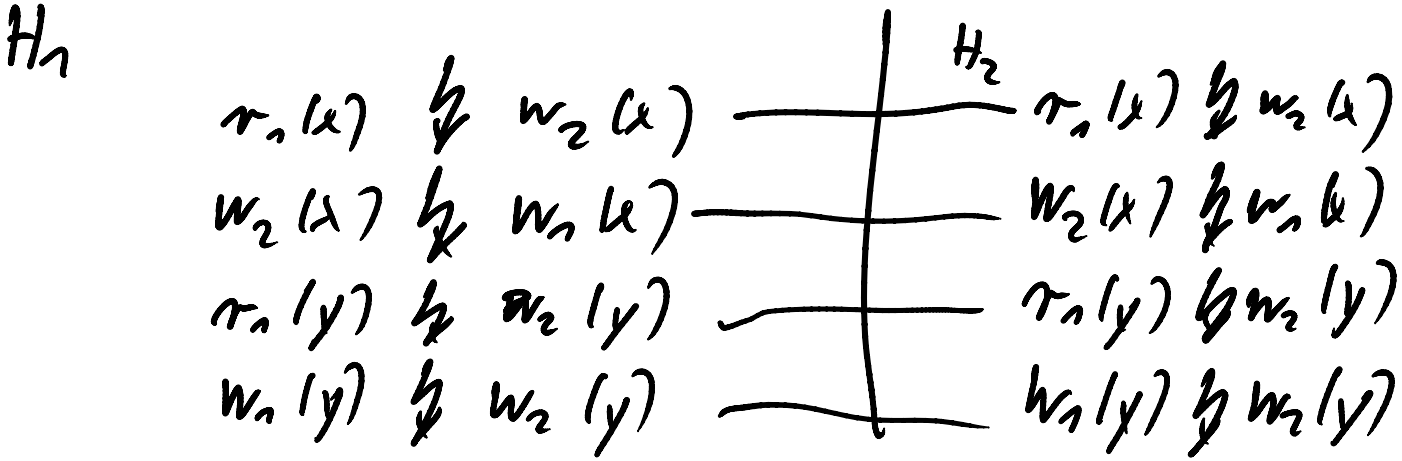
- Serielle Ausführung von Transaktionen langsam aber fehlerfrei.
- Ziel:
  - Parallele Transaktionen (Performanz)
  - Isolation (➔ ACID) gewährleisten
  - Verzahnte Ausführung von Transaktionen ohne Phänomene wie Lost Update, Dirty Read, Non-Repeatable Read, Phantom Read

Konfliktäquivalenz

- Zwei Histories H und H' sind **konflikt-äquivalent**, wenn
  - H und H' die gleichen Transaktionen bzw. Operationen enthalten,
  - die Konfliktrelationen C(H) und C(H') identisch sind.
- **Zwei Operationen p, q konfligieren** :=
  p, q greifen auf das gleiche Datenobjekt zu,
  und p oder q ist eine Schreiboperation.

Aufgabe 2

- Histories
  - $H_1 = r_1(x)r_1(y)w_2(x)w_1(y)r_2(z)w_1(x)w_2(y)c_1c_2$
  - $H_2 = r_1(y)r_1(x)w_1(y)w_2(x)w_1(x)r_2(z)w_2(y)c_1c_2$



Aufgabe 2

Test, ob $C(H_1)$ und $C(H_2)$ die gleichen Konfliktrelationen besitzen:

Konfliktrelation von (H_1):

$\{(r_1(x)<w_2(x)),(r_1(y)<w_2(y)),(w_2(x)<w_1(x)),(w_1(y)<w_2(y))\}$

Konfliktrelation von $(H_{\cancel{1}2})$:

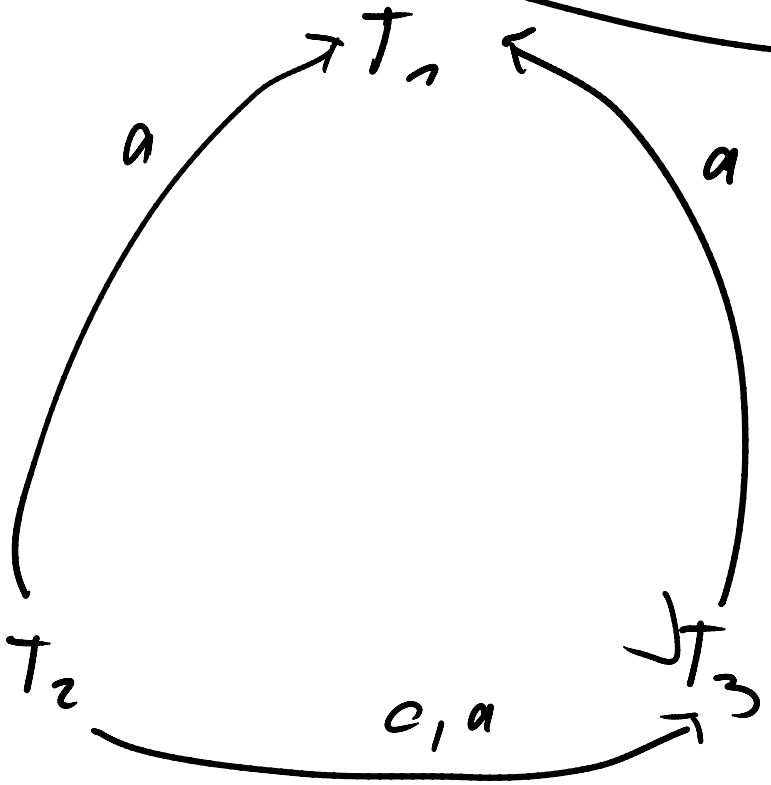
$\{(r_1(y)<w_2(y)),(r_1(x)<w_2(x)),(w_1(y)<w_2(y)),(w_2(x)<w_1(x))\}$

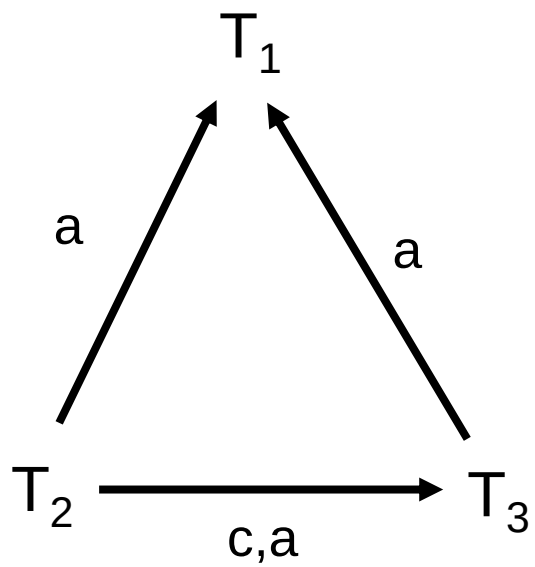
Beide Histories weisen die gleichen Konflikte auf

➔ H_1 und H_2 sind konflikt-äquivalent

Aufgabe 3: Konfliktserialisierbarkeit

$H_1 = r_3(c)r_2(b)r_1(a)w_3(c)w_1(a)c_1w_3(a)c_3r_2(c)w_2(a)w_2(c)c_2$

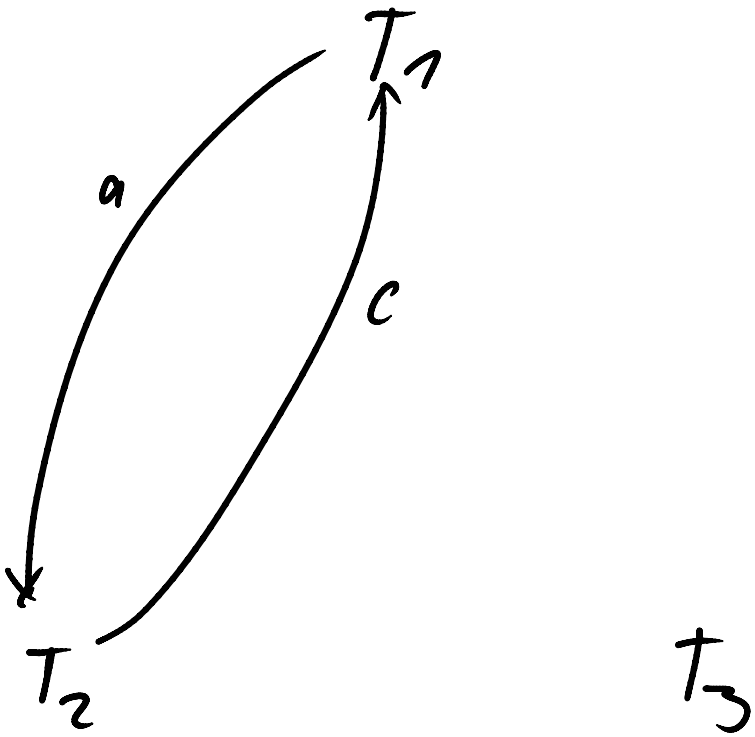


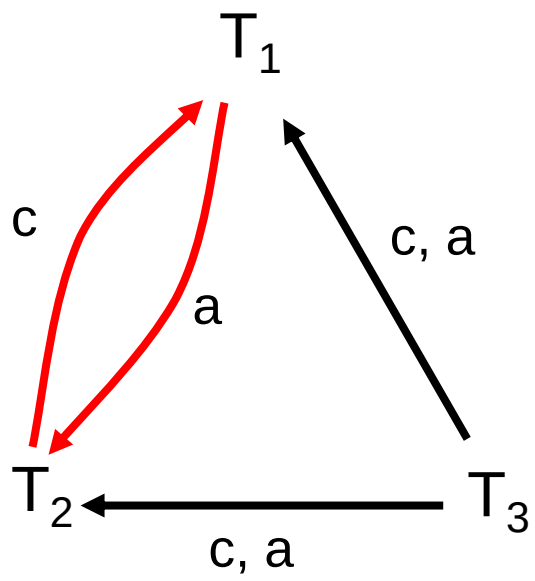


➔ $T_1T_3T_2$

Aufgabe 3: Konfliktserialisierbarkeit

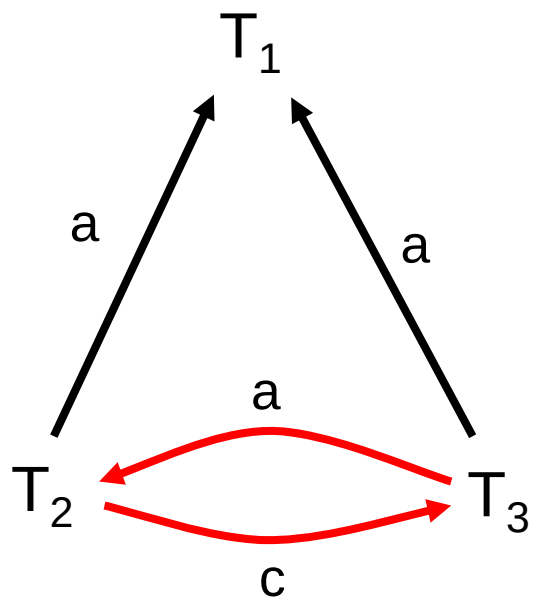
$$H_2 = r_1(c)r_2(b)r_2(c)w_2(a)w_1(a)w_2(c)r_2(c)c_2w_3(c)c_1w_3(a)c_3$$





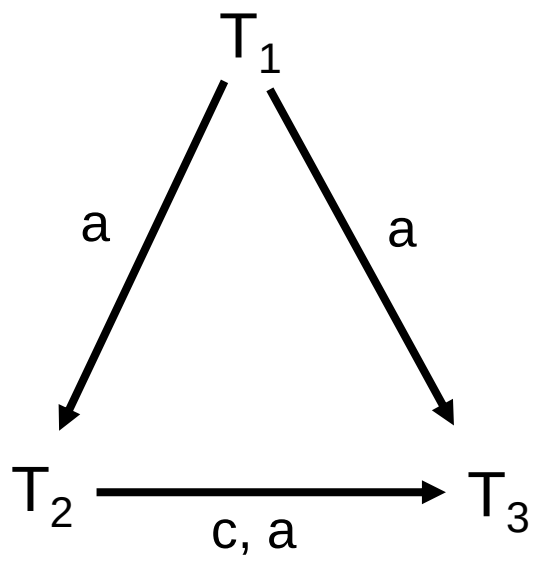
Aufgabe 3: Konfliktserialisierbarkeit

$$H_3 = r_2(b)r_3(c)w_3(c)r_1(a)r_2(c)w_1(a)c_1w_2(a)w_2(c)c_2w_3(a)c_3$$



Aufgabe 3: Konfliktserialisierbarkeit

$$H_4 = r_3(c)w_3(c)r_2(b)r_2(c)w_3(a)w_2(a)r_1(a)w_2(c)c_2c_3w_1(a)c_1$$



➔ $T_3T_2T_1$

Eigenschaften von Histories

- Conflict-Serialisability (CSR)
  - Konflikt-Serialisierbarkeit
- Recoverability (RC)
  - Rücksetzbarkeit
- Avoids Cascading Aborts (ACA)
  - keine kaskadierenden Abbrüche
- Strictness (ST)
  - keine Zwischenergebnisse von uncommitted Transaktionen lesen

Recoverability I

- ‘Commit einer Transaktion erfolgreich durchgeführt’ – kein Abort mehr möglich.
- Commit nur, wenn alle Änderungen an Daten-objekten, die T gelesen hat, committet sind.
- Gegenbeispiel: Dirty Read.

Schritt	***T1***	***T2***
1	Read(A, a1)	
2	a1 := a1-300	
3	Write(A, a1)	
4		Read(A, a2)
5		a2 := a2 *1.03
6		Write(A, a2)
7		**commit**
8	Read(B, b1)	
9	…	
10	**abort**	

Recoverability II

- *Ausführung ist recoverable* :=
  Commit von T nach Commit aller Transaktionen, von denen T gelesen hat.

- folgendes **darf nie** passieren:
  $r_1[x]\ \underline{w_1[x]\ r_{\mathbf{2}}[x]\ w_2[x]\ c_2\ a_1}$

Cascading Aborts I

- Transaktion T abortet,
  wenn sie nicht korrekt beenden kann.
- T muß zurückgesetzt werden:
  - Writes von T,
  - dto. alle anderen Transaktionen,
    die solche Änderungen gelesen haben.
    Undo kann zu *cascading abort* führen.

Cascading Aborts II

- Beispiel für cascading abort:
  - x und y haben zunächst Wert 1.
  - Zwei Transaktionen T_1 und T_2
    $r_1[x]$ $w_1[x]$ $r_2[x]$ a_1 ...
  - Abort von T_1, undo $w_1[x]$
    $\Rightarrow$ T_2 muß ebenfalls aborten.

- Cascading aborts sind möglich,
  auch wenn die History recoverable ist.

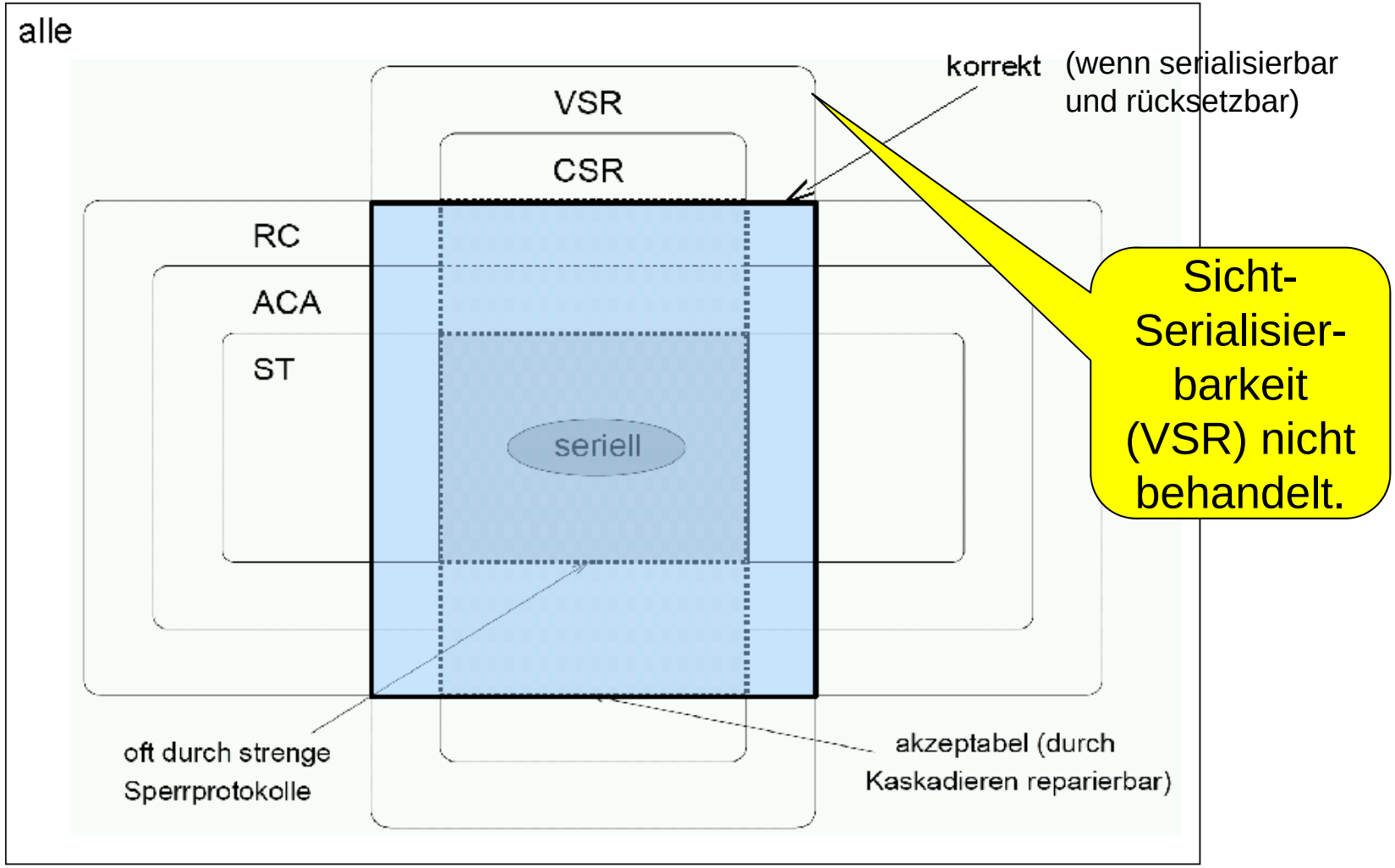
Cascadelessness

- Cascading aborts sind unerwünscht:
  - Sie ziehen ‘Buchhaltung’ nach sich,
  - Zahl der Transaktionen, die aborten müssen, ist nicht beschränkt.
- History ist *cascadeless* :=
  Jede Transaktion liest Datenobjekte nur von committeten Transaktionen.

Strictness / Reads-from-Beziehung

- ***Strictness*** := kein Wert einer nicht abgeschlossenen Transaktion darf gelesen oder überschrieben werden.
- ***Reads-from-Beziehung (rf)***: *Transaktion T_i liest von Transaktion T_j* wenn
  1. T_i liest x, nachdem T_j x geschrieben hat;
  2. T_j abortet nicht, bevor T_i x liest; und
  3. Jede Transaktion, die x schreibt, bevor T_i x liest, und nachdem T_j x überschreibt, abortet, bevor T_i x liest.

Zusammenfassung



Aufgabe 4

- $H_1 = r_3(c)r_2(b)r_1(a)w_3(c)w_1(a)c_1w_3(a)c_3r_2(c)w_2(a)w_2(c)c_2$

  < TRUE

- Recoverability (RC)
  - $T_2\ rf\ T_3\ (i \neq j) \wedge c_2 \in H \Rightarrow c_3 < c_2$ TRUE $\Rightarrow H_1 \in RC$
- Avoids Cascading Aborts (ACA) / Cascadelessness
  - $T_2\ rf\ T_3\ (i \neq j) \Rightarrow c_3 < r_2(c)$ TRUE $\Rightarrow H_1 \in ACA$
- Strictness (ST)
  - $w_3(c) < r_2(c) \Rightarrow c_3 < r_2(c)$ TRUE
  - $w_3(c) < w_2(c) \Rightarrow c_3 < w_2(c)$ TRUE
  - $w_1(a) < w_3(a) \Rightarrow c_1 < w_3(a)$ TRUE $\Rightarrow H_1 \in ST$
  - $w_1(a) < w_2(a) \Rightarrow c_1 < w_2(a)$ TRUE
  - $w_3(a) < w_2(a) \Rightarrow c_3 < w_2(a)$ TRUE

Aufgabe 4

- $H_2 = r_1(c)r_2(b)r_2(c)w_2(a)w_1(a)w_2(c)r_2(c)c_2w_3(c)c_1w_3(a)c_3$

- Recoverability (RC)
  - Keine rf-Beziehung $\Rightarrow H_2 \in RC$
- Avoids Cascading Aborts (ACA) / Cascadelessness
  - Keine rf-Beziehung $\Rightarrow H_2 \in ACA$
- Strictness (ST)
  - $w_2(a) < w_1(a) \Rightarrow c_2 < w_1(a)$ FALSE $\Rightarrow H_3 \notin ST$

Aufgabe 4

$$H_3 = r_2(b)r_3(c)w_3(c)r_1(a)r_2(c)w_1(a)c_1w_2(a)w_2(c)c_2w_3(a)c_3$$

- Recoverability (RC)
  - $T_2 \; rf \; T_3 \; (i \neq j) \wedge c_2 \in H \Rightarrow c_3 < c_2$ FALSE $\Rightarrow H_3 \notin RC$
- Avoids Cascading Aborts (ACA) / Cascadelessness
  - $H_3 \notin RC \wedge ACA \subset RC \Rightarrow H_3 \notin ACA$
- Strictness (ST)
  - $H_3 \notin ACA \wedge ST \subset ACA \Rightarrow H_3 \notin ST$

Aufgabe 4

$$H_4 = r_3(c)w_3(c)r_2(b)r_2(c)w_3(a)w_2(a)r_1(a)w_2(c)c_2c_3w_1(a)c_1$$

- Recoverability (RC)
  - $T_2 \; rf \; T_3 \; (i \neq j) \wedge c_2 \in H \Rightarrow c_3 < c_2$ FALSE $\Rightarrow H_4 \notin RC$
- Avoids Cascading Aborts (ACA) / Cascadelessness
  - $H_4 \notin RC \wedge ACA \subset RC \Rightarrow H_4 \notin ACA$
- Strictness (ST)
  - $H_4 \notin ACA \wedge ST \subset ACA \Rightarrow H_4 \notin ST$

Aufgabe 5

- **Aufgabe:** Welche der vier Histories aus Aufgabe 3 sind gemäß den oben getroffenen Aussagen korrekt? Warum?
- **Lösung:**
  - "History H_1 ist korrekt, da sie sowohl konflikt-serialisierbar als auch rücksetzbar ist."
  - "Histories H_3 und H_4 sind nicht korrekt, da sie nicht rücksetzbar sind.„
  - Keine Aussage zu Historie H_2 möglich, ohne Sicht-Serialisierbarkeit (VSR) einzuführen.

Agenda

- Informationen zu Lehrveranstaltungen am IPD im Wintersemester
- Besprechung des vierten Übungsblatts
- **Lehrevaluation**

Lehrevaluation

- Freiwilliger gesucht, der
  - Fragebögen einsammelt und
  - **heute** direkt nach der Übung bei der Evaluationsstelle der Universität abgibt (hat bis 17:30 Uhr geöffnet)