



Übung zur Vorlesung

**Informations- und
Wissensmanagement
(Übung 4)**

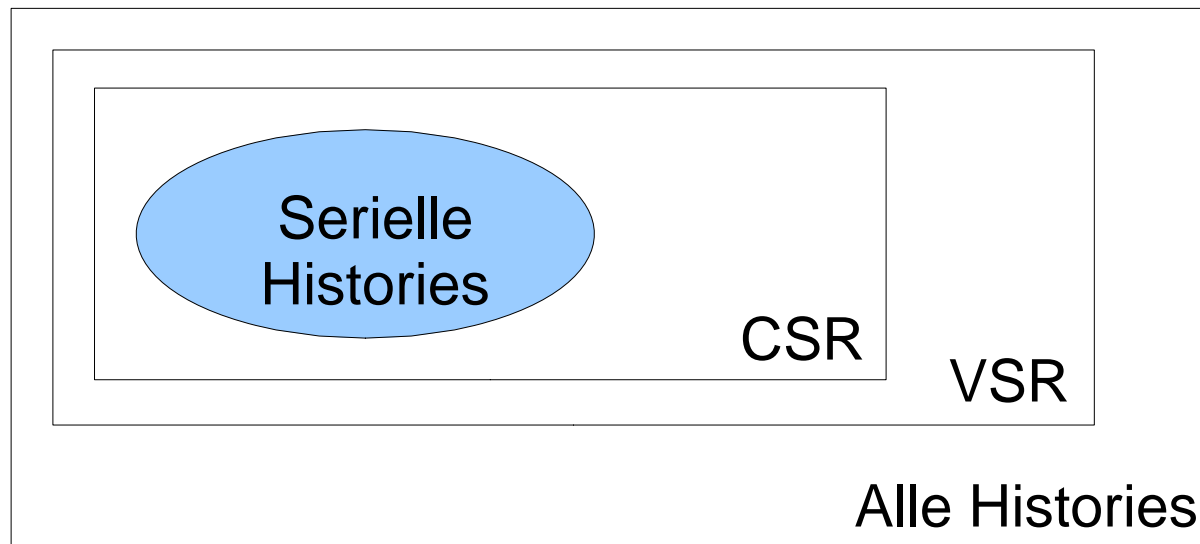
Frank Eichinger

Nebenläufigkeit & Transaktionen

- Ziel:
 - Parallele Transaktionen (Performance)
 - **ACID** gewährleisten, hier insbesondere Isolation
 - Problem: Synchronisation
 - Lost Update
 - Dirty Read
 - Non-Repeatable Read
 - Phantom
- ➔ Serialisierbarkeit

Serialisierbarkeit

- Verschiedene Definitionen
 - Serielle Histories
 - Konflikt-Serialisierbarkeit (CSR)
 - Sicht-Serialisierbarkeit (VSR)



Im Weiteren wird nur CSR betrachtet, da keine praktikablen Algorithmen für VSR existieren.

Definitionen Konfliktserialisierbarkeit

- Eine History H ist **konflikt-serialisierbar** ($H \in \text{CSR}$), wenn
 - der zugehörige Konfliktgraph azyklisch ist
 - *ODER*
 - für eine serielle History H_s gilt: $C(H)$ und $C(H_s)$ sind konflikt-äquivalent
- Zwei Histories H und H' sind **konflikt-äquivalent**, wenn
 - H und H' die gleichen Transaktionen bzw. Operationen enthalten
 - die Konfliktrelationen von $C(H)$ und $C(H')$ identisch sind

Rücksetzbarkeit (RC + ACA)

- **Rücksetzbar (RC - recoverable):**
 - $T_i \text{ rff } T_j (i \neq j) \wedge c_i \in H \Rightarrow c_j < c_i$
 - Eine Transaktion T_i darf erst dann freigegeben (committed, erfolgreich abgeschlossen) werden, wenn alle Transaktionen T_j , von denen sie gelesen hat, bereits freigegeben sind.
- **Vermeidung kaskadierender Abbrüche (ACA - avoiding cascading aborts):**
 - $T_i \text{ rff } T_j (i \neq j) \Rightarrow c_j < r_i(x)$
 - Jede Transaktion T_i liest nur Datenobjekte von bereits freigegebenen Transaktionen T_j .

Rücksetzbarkeit (ST)

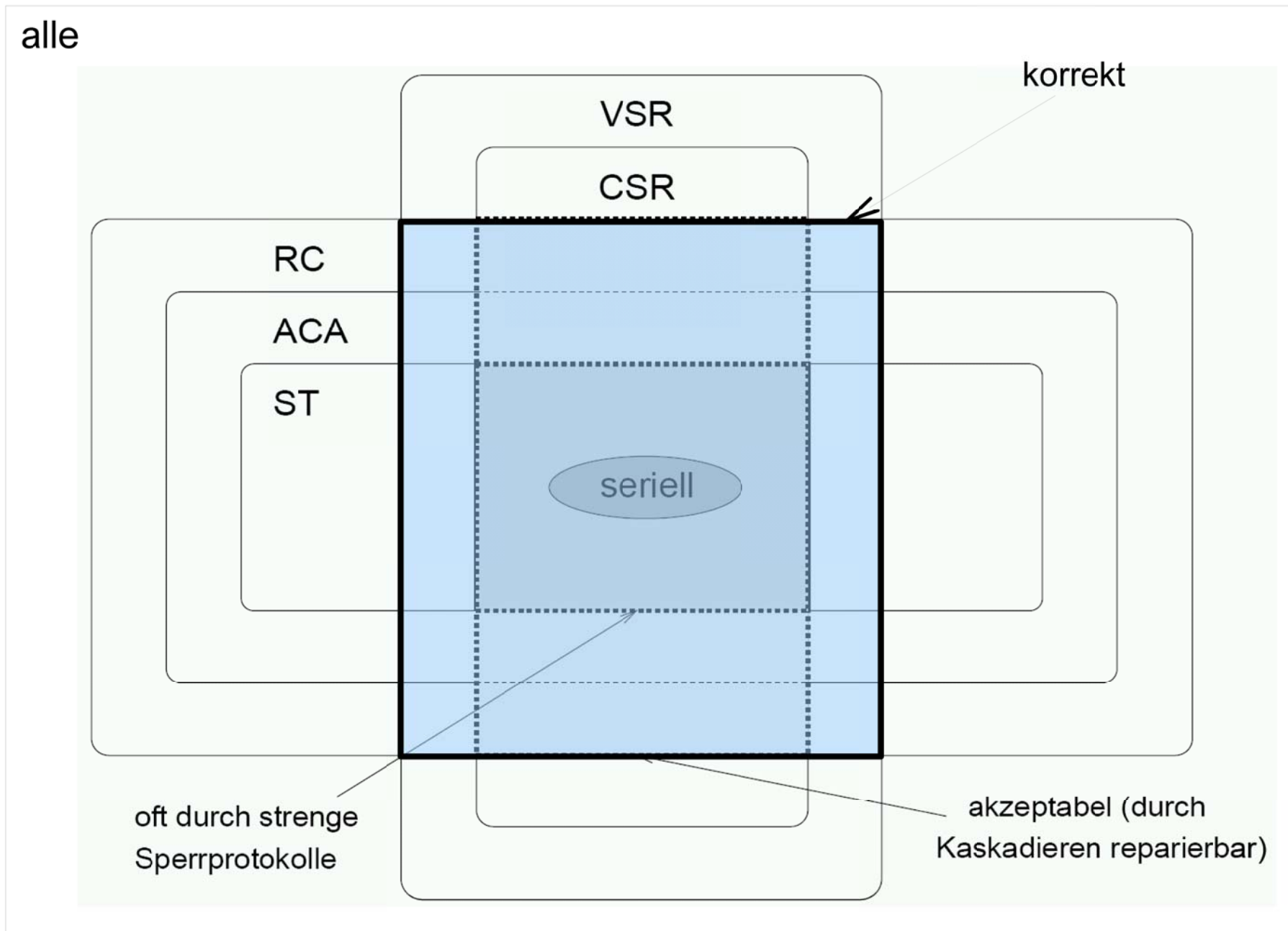
- Probleme mit Before-Images

DB-Inhalt	Operation
$x = 1$ (Anfangswert)	
$x = 2$	$w_1(x \leftarrow 2)$ [$BF_{x,T_1} = 1$]
$x = 3$	$w_2(x \leftarrow 3)$ [$BF_{x,T_2} = 2$]
	a_1 Rücksetzen von $w_1(x \leftarrow 2)$ mit $BF_x := 1$. Überschreiben durch T_2 muß erhalten bleiben!
$x = 3$	a_2 Rücksetzen von $w_2(x \leftarrow 3)$ mit $BF_x := ??$

- Striktheit (ST):

- $w_j(x) < o_i(x)$ ($i \neq j$) $\Rightarrow c_j < o_i(x) \vee a_j < o_i(x)$
- Es darf kein „geschriebenes“ Objekt einer noch nicht beendeten (also noch laufenden) Transaktion gelesen oder überschrieben werden.

Zusammenfassung



Aufgabe 1

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

$$H_1 = r_1(x) r_1(y) w_2(x) w_1(y) r_2(z) w_1(x) w_2(y) c_1 c_2$$

$$H_2 = r_1(y) r_1(x) w_1(y) w_2(x) w_1(x) r_2(z) w_2(y) c_1 c_2$$

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_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 2a)

Bestimmen Sie, welche der Histories
konflikt-serialisierbar (CSR) sind.

$$H_1 = r_3(c) r_2(b) r_1(a) w_3(c) w_1(a) c_1 w_3(a) c_3 r_2(c) w_2(a) w_2(c) c_2$$

$$H_2 = r_1(c) r_2(b) r_2(c) w_2(a) w_1(a) w_2(c) r_3(c) c_2 w_3(c) c_1 w_3(a) c_3$$

$$H_3 = r_2(b) r_3(c) w_3(c) r_1(a) r_2(c) w_1(a) c_1 w_2(a) w_2(c) c_2 w_3(a) c_3$$

$$H_4 = r_3(c) w_3(c) r_2(b) r_2(c) w_3(a) w_2(a) r_1(a) w_2(c) c_2 c_3 w_1(a) c_1$$

Aufgabe 2b)

- Weitere Aussagen:
 - H1 und H4 sind konflikt-serialisierbar (CSR), damit sind sie auch sicht-serialisierbar (VSR).
 - H2 und H3 sind nicht konflikt-serialisierbar (CSR), damit kann aber nicht ausgeschlossen werden, dass sie auch nicht sicht-serialisierbar (VSR) sind.

Aufgabe 3

- $H_1 = r_3(c) r_2(b) r_1(a) w_3(c) w_1(a) c_1 w_3(a) c_3 r_2(c) w_2(a) w_2(c) c_2$

$T_2 \text{ rf } T_3 (i \neq j) \wedge c_2 \in H \Rightarrow c_3 < c_2$ TRUE $\Rightarrow H_1 \in RC$

$T_2 \text{ rf } T_3 (i \neq j) \Rightarrow c_3 < r_2(c)$ TRUE $\Rightarrow H_1 \in ACA$

$w_3(c) < r_2(c) \Rightarrow c_3 < r_2(c)$ 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

- $H_2 = r_1(c) r_2(b) r_2(c) w_2(a) w_1(a) w_2(c) r_3(c) c_2 w_3(c) c_1 w_3(a) c_3$

$T_3 \text{ rf } T_2 (i \neq j) \wedge c_3 \in H \Rightarrow c_2 < c_3$ TRUE $\Rightarrow H_2 \in RC$

$T_3 \text{ rf } T_2 (i \neq j) \Rightarrow c_2 < r_3(c)$ FALSE $\Rightarrow H_2 \notin ACA$

$H_2 \notin ACA \wedge ST \subset ACA$ $\Rightarrow H_2 \notin ST$

Aufgabe 3

- $H_3 = r_2(b) r_3(c) w_3(c) r_1(a) r_2(c) w_1(a) c_1 w_2(a) w_2(c) c_2 w_3(a) c_3$
 $T_2 \text{ rf } T_3 (i \neq j) \wedge c_2 \in H \Rightarrow c_3 < c_2$ FALSE $\Rightarrow H_3 \notin RC$
 $H_3 \notin RC \wedge ACA \subset RC \Rightarrow H_3 \notin ACA$
 $H_3 \notin ACA \wedge ST \subset ACA \Rightarrow H_3 \notin ST$
- $H_4 = r_3(c) w_3(c) r_2(b) r_2(c) w_3(a) w_2(a) r_1(a) w_2(c) c_2 c_3 w_1(a) c_1$
 $T_2 \text{ rf } T_3 (i \neq j) \wedge c_2 \in H \Rightarrow c_3 < c_2$ FALSE $\Rightarrow H_4 \notin RC$
 $T_1 \text{ rf } T_2 (i \neq j) \wedge c_1 \in H \Rightarrow c_2 < c_1$ TRUE
 $H_4 \notin RC \wedge ACA \subset RC \Rightarrow H_4 \notin ACA$
 $H_4 \notin ACA \wedge ST \subset ACA \Rightarrow H_4 \notin ST$

Aufgabe 4

- Eindeutige Aussagen:
 - "History H1 ist korrekt, da sie sowohl serialisierbar (CSR) als auch rücksetzbar (RC) ist."
 - "Histories H3 und H4 sind nicht korrekt, da sie beide nicht rücksetzbar (RC) sind."
- Offen:
 - History H2 ist rücksetzbar (RC) aber nicht in CSR. H2 ist dann korrekt, wenn die Sicht-Serialisierbarkeit (VSR) gezeigt werden kann.

Aufgabe 5a)

```
public void insertNewCD(String title, integer year)
{
    try {
        // SQL-Anweisung erstellen
        String sqlStatement =
            "INSERT INTO CD VALUES ("
            + title + ", " + year + ")";

        // Verbindung zur DB herstellen
        Class.forName("...");
        Connection con =
            DriverManager.getConnection("...");

        // Statement erzeugen
        Statement statement = con.createStatement();

        // SQL-Anweisung absetzen
        statement.executeUpdate(sqlStatement);
    } catch (Exception e) {
        System.out.println("Fehler beim Einfügen.");
    }
}
```

Aufgabe 5b)

```
public void deleteVideoClip(integer id) {
    try {
        // SQL-Anweisung erstellen
        String sqlStatement =
            "DELETE FROM VideoClip WHERE ID = " + id;

        // Verbindung zur DB herstellen
        Class.forName("...");
        Connection con =
            DriverManager.getConnection("...");

        // Statement erzeugen
        Statement statement = con.createStatement();

        // SQL-Anweisung absetzen
        statement.executeUpdate(sqlStatement);
    } catch (Exception e) {
        System.out.println("Fehler beim Löschen.");
    }
}
```

Aufgabe 5c)

```
public void getAllCDsByYear(integer year) {
    try {
        // SQL-Anweisung erstellen
        String sqlStatement =
            "SELECT * FROM CD WHERE Jahr = " + year;

        // Verbindung zur DB herstellen
        Class.forName("...");
        Connection con = DriverManager.getConnection("...");

        // Statement erzeugen
        Statement statement = con.createStatement();

        // SQL-Anweisung absetzen
        ResultSet res = statement.executeQuery(sqlStatement);

        // Ergebnisse ausgeben (Titel, Jahr)
        while (res.next()) {
            System.out.println(res.getString(1) + ", " +
                res.getInteger(2));
        }
    } catch (Exception e) {
        System.out.println("Fehler beim DB-Zugriff.");
    }
}
```

Aufgabe 6

