Ludwig-Maximilians-Universität München
Department "Institut für Informatik"
Prof. Dr. Christian Böhm
Bianca Wackersreuther

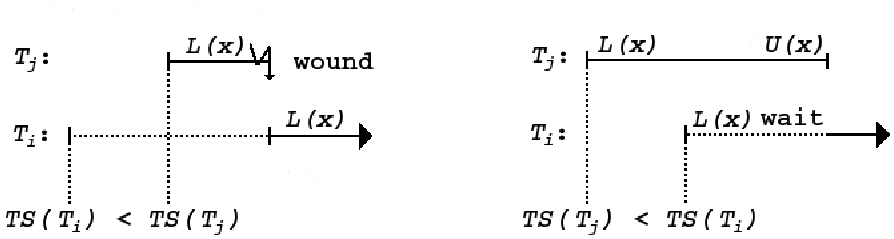
Datenbanksysteme II

SS 2007

Übungsblatt 2: Synchronisationsverfahren

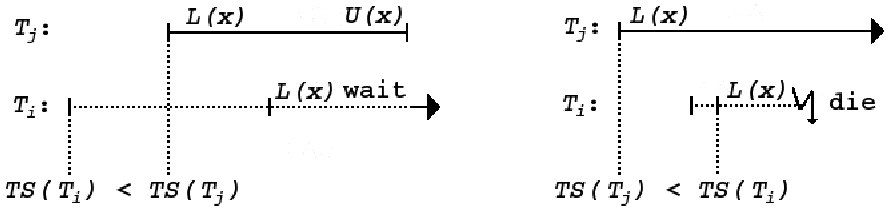
- **Sperrprotokolle**:
  - RX-Protokoll
  - RUX-Protokoll
  - RAX-Protokoll
  - RIX-Protokoll
- **Zeitstempelverfahren (Wound-Wait)**: jüngere Transaktionen warten auf ältere Transaktionen, sonst rücksetzen

  T_i fordert Sperre $L(x)$ an.
  - jüngere TA T_j, d.h. $TS(T_j) > TS(T_i)$, hält bereits Sperre auf x:
    $\Rightarrow T_i$ läuft weiter, jüngere TA T_j wird zurückgesetzt (*Wound*)
  - ältere TA T_j, d.h. $TS(T_j) < TS(T_i)$, hält bereits Sperre auf x:
    $\Rightarrow T_i$ wartet auf Freigabe der Sperre durch ältere TA T_j (*Wait*)



- **Zeitstempelverfahren (Wait-Die)**: ältere Transaktionen warten auf jüngere Transaktionen, sonst rücksetzen

  T_i fordert Sperre $L(x)$ an.
  - jüngere TA T_j, d.h. $TS(T_j) > TS(T_i)$, hält bereits Sperre auf x:
    $\Rightarrow T_i$ wartet auf Freigabe der Sperre durch jüngere TA T_j (*Wait*)
  - ältere TA T_j, d.h. $TS(T_j) < TS(T_i)$, hält bereits Sperre auf x:
    $\Rightarrow T_i$ wird zurückgesetzt (*Die*), ältere TA T_j läuft weiter



- **Synchronisation ohne Sperren: "Zeitstempel statt Sperren auf Objekten"**
  - Nicht nur Transaktionen, sondern auch Objekte O tragen Zeitstempel:
    - $readTS(O)$: Zeitstempel der jüngsten TA, die das Objekt O gelesen hat
    - $writeTS(O)$: Zeitstempel der jüngsten TA, die das Objekt O geschrieben hat
  - Prüfungen beim *Lesezugriff* von T_i auf ein Objekt O:
    - $\mathbf{L}_1$: Falls $TS(T_i) < writeTS(O)$: T_i ist älter als die TA, die O geschrieben hat $\Rightarrow T_i$ zurücksetzen (und Zeitstempel zurücksetzen)
    - $\mathbf{L}_2$: Falls $TS(T_i) \geq writeTS(O)$: T_i darf O lesen, Lesemarke wird aktualisiert: $readTS(O) = max(TS(T_i, readTS(O)))$
  - Prüfungen beim *Schreibzugriff* von T_i auf ein Objekt O:
    - $\mathbf{S}_1$: Falls $TS(T_i) < readTS(O)$: T_i ist älter als die TA, die O gerade gelesen hat $\Rightarrow T_i$ zurücksetzen
    - $\mathbf{S}_2$: Falls $TS(T_i) < writeTS(O)$: T_i ist älter als die TA, die O geschrieben hat $\Rightarrow T_i$ zurücksetzen
    - $\mathbf{S}_3$: Sonst: T_i darf O schreiben, Schreibmarke wird aktualisiert: $writeTS(O) = TS(T_i)$

- **Optimistische Synchronisation**:
  - Für jede Transaktion T_i werden zwei Mengen geführt:
    - $RS(T_i)$: die von T_i gelesenen Objekte (*Read Set*)
    - $WS(T_i)$: die von T_i geschriebenen Objekte (*Write Set*)
  - Zwei Validierungsstrategien:
    - *Backward-Oriented Optimistic Concurrency Control* (BOCC):
      Validierung nur gegenüber bereits beendeten TAs
      Algorithmus:

```
VALID = true;
for (alle während Ausführung von T_i beendeten T_j) do
                if RS(T_i) ∩ WS(T_j) ≠ ∅ then VALID = false;
end;
if VALID then Schreibphase(T_i) else Rollback(T_i);
```

    - *Forward-Oriented Optimistic Concurrency Control* (FOCC):
      Validierung nur gegenüber noch laufenden TAs
      Algorithmus:

```
VALID = true;
for (alle laufenden T_j) do
                if WS(T_i) ∩ RS(T_j) ≠ ∅ then VALID = false;
end;
if VALID then Schreibphase(T_i) else löse Konflikt auf;
```