Datenbanksysteme

Transaktionen

Burkhardt Renz

Fachbereich MNI
Technische Hochschule Mittelhessen

Sommersemester 2020

Übersicht

- Transaktionen
  - Motivation
  - ACID-Eigenschaften
- Recovery
  - Ursachen für Recovery
  - Logging und Backup
  - Vorgehen bei Plattencrash
- Isolationslevel
  - Phänomene
  - Isolationslevel

Beispiel 1

Wir wollen eine Überweisung durchführen von Konto 1 auf Konto 2.

Die Daten der Konten werden in der Tabelle Konto aufgezeichnet:

KtoNr	**Saldo**
1	100
2	100

```
update Konto set Saldo = Saldo - 50 where KtoNr = 1;
update Konto set Saldo = Saldo + 50 where KtoNr = 2;
```

Was passiert, wenn der Rechner mittendrin abstürzt??

Beispiel 1 – mit Transaktion

Wir wollen eine Überweisung durchführen von Konto 1 auf Konto 2.

Die Daten der Konten werden in der Tabelle Konto aufgezeichnet:

KtoNr	**Saldo**
1	100
2	100

```
begin transaction;
  update Konto set Saldo = Saldo - 50 where KtoNr = 1;
  update Konto set Saldo = Saldo + 50 where KtoNr = 2;
commit;
```

Was passiert, wenn der Rechner mittendrin abstürzt??

Beispiel 2

Wir wollen eine Überweisung durchführen von Konto 1 auf Konto 2. Sie soll nur möglich sein, wenn das Konto nicht überzogen wird.

Die Daten der Konten werden in der Tabelle Konto aufgezeichnet:

KtoNr	**Saldo**
1	100
2	100

```
select Saldo from Konto where KtoNr = 1;
if (Saldo > 60) {
  update Konto set Saldo = Saldo - 60 where KtoNr = 1;
  update Konto set Saldo = Saldo + 60 where KtoNr = 2;
}
```

Was passiert, wenn zwischendrin eine andere Überweisung von Kto 1 stattfindet??

Beispiel 2 – mit Transaktion

Wir wollen eine Überweisung durchführen von Konto 1 auf Konto 2. Sie soll nur möglich sein, wenn das Konto nicht überzogen wird.

Die Daten der Konten werden in der Tabelle Konto aufgezeichnet:

KtoNr	**Saldo**
1	100
2	100

```
begin transaction isolation level serializable;
  select Saldo from Konto where KtoNr = 1;
  if (Saldo > 60) {
    update Konto set Saldo = Saldo - 60 where KtoNr = 1;
    update Konto set Saldo = Saldo + 60 where KtoNr = 2;
  }
commit;
```

Was passiert, wenn zwischendrin eine andere Überweisung von Kto 1 stattfindet??

Transaktion

Definition

Eine Transaktion ist eine logische Verarbeitungseinheit auf einer Datenbank, die eine oder mehrere Datenbankoperationen (Einfügen, Löschen, Ändern und/oder Suchen) umfasst.

Eine Transaktion wird mit dem Befehl `commit` als gültig erklärt oder mit dem Befehl `rollback` rückgängig gemacht.

Eigenschaften von Transaktionen

A	Atomarität (*atomicity*)
C	Konsistenz (*consistency*)
I	Isolation (*isolation*)
D	Dauerhaftigkeit (*durability*)

Atomarität

Definition (Atomarität)

Die Teilschritte einer Transaktion werden vom DBMS als eine unteilbare, atomare Einheit durchgeführt, d.h. alle oder gar keiner.

Merke: „Alles oder nichts“

Konsistenz

Definition (Konsistenz)

Die Datenbank hat vor und nach einer Transaktion stets einen konsistenten Zustand, d.h. alle Integritätsbedingungen des Datenbankschemas sind erfüllt.

Merke: „Daten bleiben konsistent“

Isolation

Definition (Isolation)

Eine Transaktion läuft isoliert gegenüber dem Einfluss anderer Transaktionen ab, so als ob sie exklusiven Zugriff auf die Daten hätte.

Eventuell wird die Transaktion vom DBMS abgebrochen, weil andernfalls ein unerlaubter Einfluss anderer Transaktionen erfolgt wäre.

Merke: „Eine Transaktion hat die Daten quasi allein“

Dauerhaftigkeit

Definition (Dauerhaftigkeit)

Die Ergebnisse einer bestätigten Transaktion (nach dem Commit) sind dauerhaft gesichert, d.h. das DBMS garantiert, dass bei einem Fehler der bestätigte Zustand wiederhergestellt werden kann.

Merke: „Nichts geht verloren“

Übersicht

- Transaktionen
  - Motivation
  - ACID-Eigenschaften
- Recovery
  - Ursachen für Recovery
  - Logging und Backup
  - Vorgehen bei Plattencrash
- Isolationslevel
  - Phänomene
  - Isolationslevel

Worum geht es beim Recovery?

Atomarität verlangt, dass eine Transaktion vollständig oder gar nicht ausgeführt wird.

Konsistenz verlangt, dass eine Datenbank auch nach einem Fehler in einem konsistenten Zustand ist.

Dauerhaftigkeit verlangt, dass einmal bestätigte Änderungen dauerhaft erhalten bleiben.

Definition

Mit Recovery bezeichnet man die Mechanismen eines DBMS um auch nach einem Fehler oder dem Fehlschlagen einer Transaktion einen ordnungsgemäßen Betrieb in einem konsistenten Zustand zu garantieren.

Abbruch einer Transaktion

Eine Transaktion kann abgebrochen werden durch

- *Benutzerabbruch* – wenn er während einer Transaktion interaktiv Zugriff hat
- *Ausnahmebedingungen* – Prüfungen innerhalb der Transaktion, die zu einem Abbruch führen
- *Nebenläufigkeitskontrolle* – etwa wenn das DBMS eine Verklemmung oder eine Verletzung der Serialisierbarkeit entdeckt
- *Systemfehler* – etwa bei einer Division durch Null in einer Transaktion

Schwere Fehler

Das DBMS kann abstürzen durch

- *Systemabsturz* – etwa wegen Stromausfall, Netzausfall, Softwarefehler in der Software des DBMS, Datenfehler
- *Hardware-Ausfall* – etwa Plattenfehler, dies betrifft dann nicht nur laufende Transaktionen, sondern auch ältere Daten

Grundlegendes Konzept

Um verlorene Daten durch ein Recovery wieder herstellen zu können, braucht man *Redundanz*.

Zwei wichtige Formen:

- *Logging* – Der Ablauf von Transaktionen wird laufend mitprotokolliert
- *Backup* – Der Zustand der Datenbank wird dauerhaft auf einem besonderen Medium gesichert

Write-Ahead Log

Definition

Ein Systemlog, kurz Log ist eine Folge von Einträgen, die den Ablauf und den Inhalt von Transaktionen protokollieren.

Definition

Ein Write-Ahead Log ist eine Technik des Protokolls von Transaktionen, bei der immer zunächst das Log geschrieben wird und erst dann Änderungen an den Nutzdaten vorgenommen werden.

Arten der Einträge im Log

Daten werden in die Datenbasis immer *blockweise* geschrieben – in Speicherblöcken zu z.B. 8 KiloByte.

Definition

Ein before image ist der Zustand des von einer Transaktion veränderten Blocks *vor* der Änderung.

Möchte man Änderungen rückgängig machen (undo), benötigt man das before image.

Definition

Ein after image ist der Zustand des von einer Transaktion veränderten Blocks *nach* der Änderung.

Möchte man Änderungen wiederholen (redo), benötigt man das after image.

Einträge im Log

- $[T_n, \texttt{begin}]$ – Beginn der Transaktion mit der Nummer n
- $[T_n, \texttt{update, before-image, after-image}]$ – Eintrag über Änderung eines Blocks innerhalb der Transaktion mit der Nummer n
- $[T_n, \texttt{commit}]$ – Ende der Transaktion mit commit
- $[T_n, \texttt{abort}]$ – Ende der Transaktion mit Rollback
- $[\texttt{checkpoint}, T_i, T_j, T_n]$ – Checkpoint mit Nummern der noch aktiven Transaktionen.

Vorgehen beim Rollback

Beispiel einer Situation im Log beim Rollback – Die Transaktion T_2 wird abgebrochen.

Log zum Zeitpunkt des Rollbacks:

1: $[T_2, \texttt{begin}]$
2: $[T_3, \texttt{begin}]$
3: $[T_3, \texttt{update, before-image}]$
4: $[T_2, \texttt{update, before-image}]$
5: $[T_3, \texttt{update, before-image}]$

Vorgehen beim Systemcrash

...
10:$[T_2, \texttt{begin}]$
11:$[T_3, \texttt{begin}]$
12:$[T_2, \texttt{update, before-image, after-image}]$
13:$[T_1, \texttt{begin}]$
14:$[T_2, \texttt{commit}]$
15:$[T_5, \texttt{begin}]$
16:$[T_3, \texttt{update, before-image, after-image}]$
17:$[T_5, \texttt{update, before-image, after-image}]$
18:$[T_5, \texttt{abort}]$
19:$[\texttt{checkpoint}\, T_1, T_3, T_4]$
20:$[T_1, \texttt{update, before-image, after-image}]$
21:$[T_4, \texttt{update, before-image, after-image}]$
22:$[T_6, \texttt{begin}]$
23:$[T_4, \texttt{commit}]$
24:$[T_6, \texttt{update, before-image, after-image}]$
25:$[T_1, \texttt{update, before-image, after-image}]$

Backup

Hardware

- Trennung von Log-Dateien und Nutzdaten auf verschiedene Platten
- Geeignete Plattensysteme, z.B. RAID

Datensicherungen

- *Offline*-Backup
- *Online*-Backup – im laufenden Betrieb, Daten + Logs

Beispiel Oracle

Backup

umfasst alle Datenfiles und das Controlfile für eine Oracle Datenbank

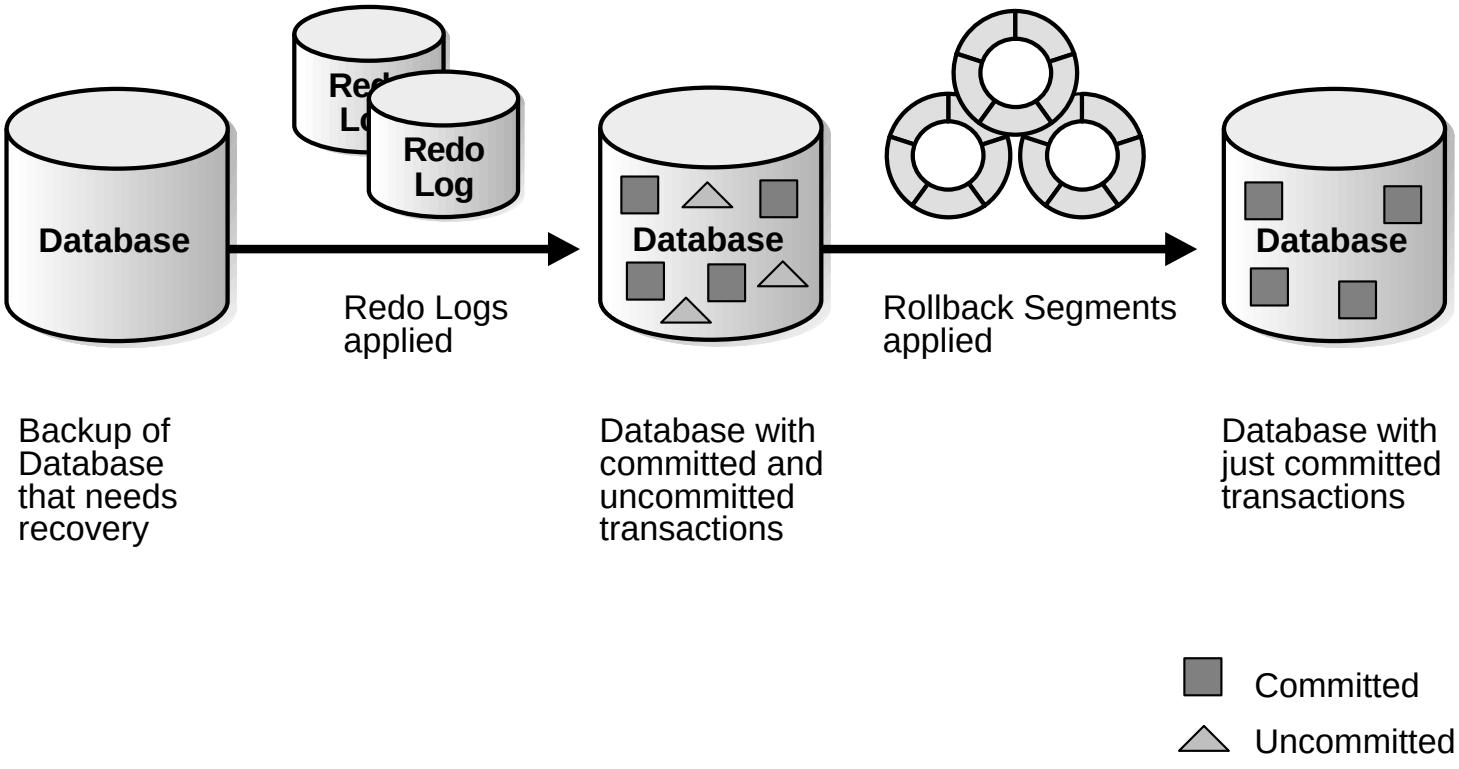
Redolog

Log mit allen Änderungen an den Datenfiles, gespeichert auf einem Medium getrennt von den Datenfiles

Rollback Segments

Speichert die Undo-Logs für laufende Transaktionen

Beispiel Oracle



Übersicht

- Transaktionen
  - Motivation
  - ACID-Eigenschaften
- Recovery
  - Ursachen für Recovery
  - Logging und Backup
  - Vorgehen bei Plattencrash
- Isolationslevel
  - Phänomene
  - Isolationslevel

Serielle und serialisierbare Abläufe

Definition (Serieller Ablauf)

Ein Ablauf von Transaktionen mit ihren Teilschritten heißt seriell, wenn alle Teilschritte einer Transaktion vollständig ausgeführt werden, ehe die der nächsten Transaktion beginnen.

Definition (Serialisierbarer Ablauf)

Ein Ablauf von Transaktionen mit ihren Teilschritten heißt serialisierbar, wenn er im Effekt äquivalent zu einem seriellen Ablauf ist.
D.h. die Transaktionen sind zwar mit ihren Teilschritten verschränkt, kommen aber nicht in Konflikt zueinander.

Isolation

- Die ACID-Prinzipien schreiben für eine Transaktion Isolation vor, d.h. eine Transaktion darf nicht durch Aktionen anderer Transaktionen beeinflußt werden, also Serialisierbarkeit.
- Manchmal braucht man jedoch nicht eine so strenge Isolation. Der SQL-Standard sieht deshalb Isolationslevel vor – Grade der Abschirmung von Transaktionen.
- Der Standard definiert bestimmte Phänomene der Beeinflussung und dann die Isolationslevel dadurch, inwiefern solche Phänomene vorkommen können.

Lost Update[1]/Dirty Write

T_1	T_2	Saldo
		100

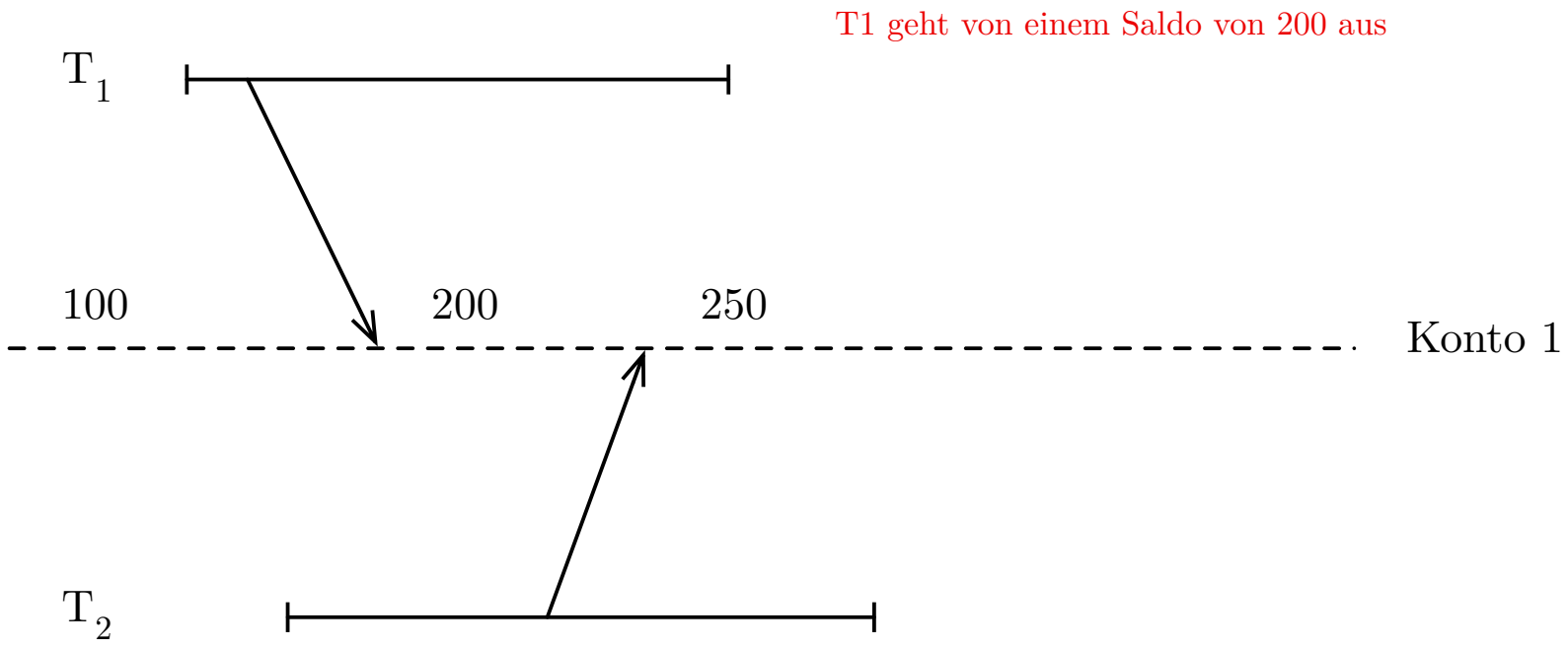
`update Konto`
`set Saldo = 200`
`where KtoNr = 1`
		200

	`update Konto`
	`set Saldo = 250`
	`where KtoNr = 1`
		250

`commit`		200
	`commit`	250

Die Änderung von Transaktion T_2 hat die Änderung von Transaktion T_1 überschrieben.

Dirty Write



Dirty Read

T_1	T_2	Saldo für Konto 1
		100

	`update Konto`
	`set Saldo = 200`
	`where KtoNr = 1`
		200

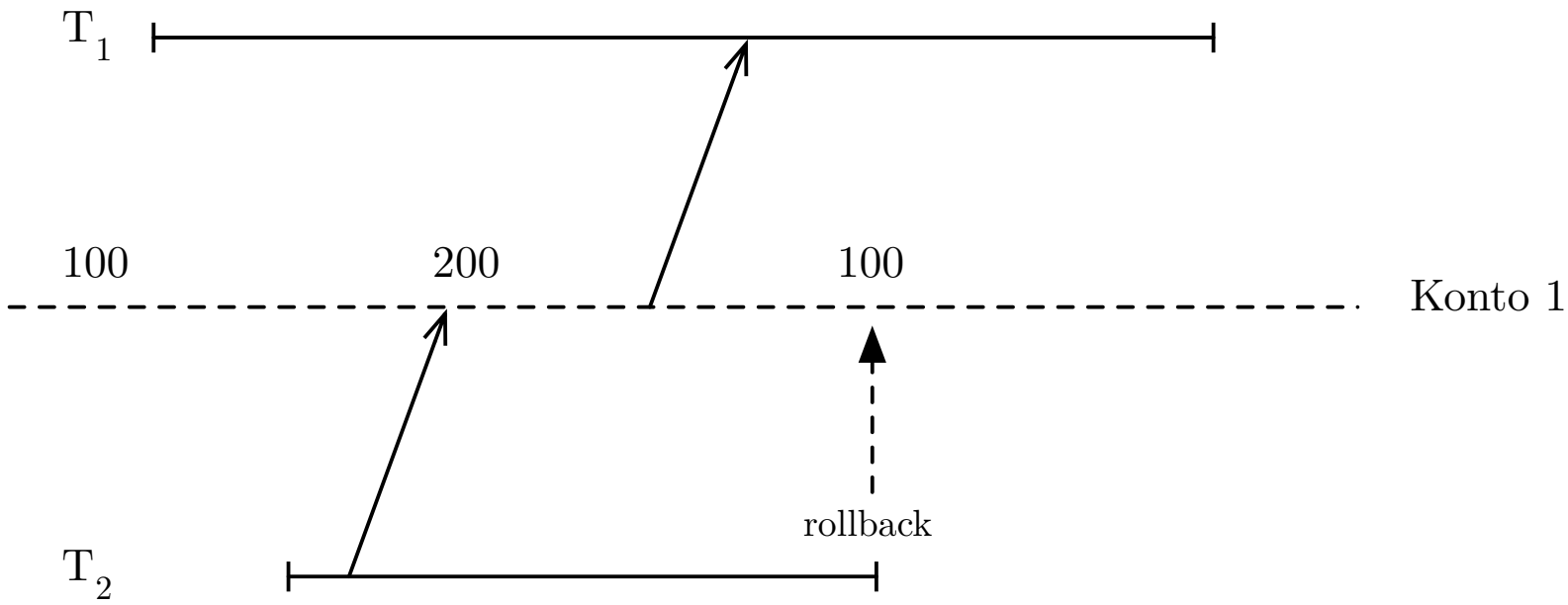
`select Saldo`
`from Konto`
`where KtoNr = 1`
...
T_1 verwendet den Wert 200		

	`rollback`	100
`commit`		100

Transaktion T_1 hat einen Wert verwendet, der niemals gültig war!

Dirty Read

T1 geht von einem Saldo von 200 aus



Non-repeatable Read

T_1	T_2	Saldo für Konto 1
		100

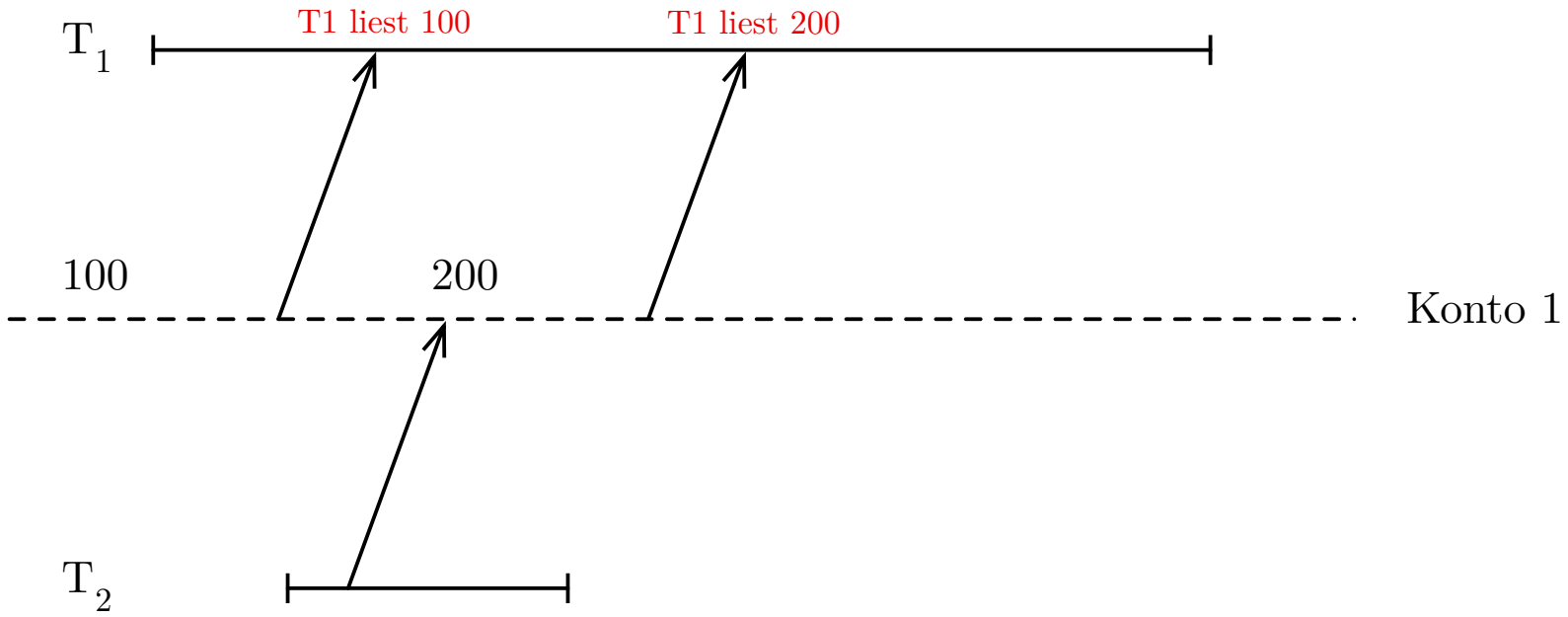
`select Saldo`
`from Konto`
`where KtoNr = 1`
T_1 liest den Wert 100		

	`update Konto`
	`set Saldo = 200`
	`where KtoNr = 1`
	`...`
	`commit`
		200

`select Saldo`
`from Konto`
`where KtoNr = 1`
T_1 liest den Wert 200		

Transaktion T_1 kann sich nicht darauf verlassen, dass ein gelesener Wert gültig bleibt!

Non-repeatable Read



Falsche Ergebnisse bei Non-repeatable Read

T_1	T_2	Saldo Konten 1–3
		`40, 50, 30` Summe: 120
T_1 liest Saldo von Konto 1 und schreibt den Wert in `sum`, also 40.		
	T_2 ändert Saldo von Konto 3 auf 20 und von Konto 1 auf 60 `commit`	`60, 50, 20` Summe: 130
T_1 liest Saldo von Konto 2 und addiert den Wert zu `sum`, also 90.		
T_1 liest Saldo von Konto 3 und addiert den Wert zu `sum`, also 110.		

Phantom Row

T_1	T_2	Saldo für Konten
		`100, 100`

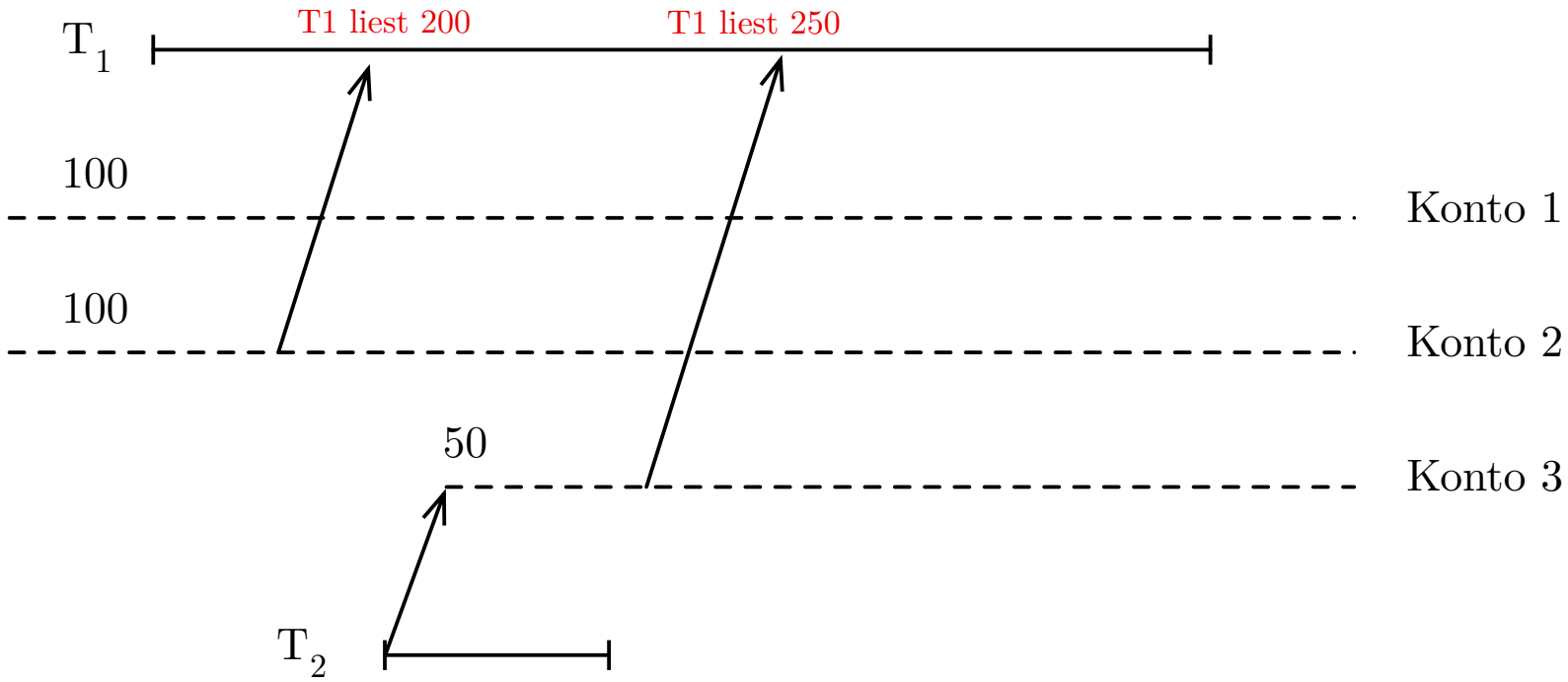
`select sum(Saldo)`
`from Konto`
T_1 liest den Wert 200		

	`insert into Konto`
	`values (3,50)`
	`commit`
		`100, 100, 50`

`select sum(Saldo)`
`from Konto`
T_1 liest den Wert 250		

Transaktion T_1 bekommt Datensätze „untergeschoben“!

Phantom Row


T1 liest 200
T1 liest 250
T1
100
Konto 1
100
Konto 2
50
Konto 3
T2

Definition der Isolationslevel in SQL

	Dirty Read	Non-Repeatable Read	Phantom Row
`READ UNCOMMITTED`	möglich	möglich	möglich
`READ COMMITTED`	*nicht* möglich	möglich	möglich
`REPEATABLE READ`	*nicht* möglich	*nicht* möglich	möglich
`SERIALIZABLE`	*nicht* möglich	*nicht* möglich	*nicht* möglich

Beispiel

Wir wollen eine Überweisung durchführen von Konto 1 auf Konto 2. Sie soll nur möglich sein, wenn das Konto nicht überzogen wird.

Die Daten der Konten werden in der Tabelle Konto aufgezeichnet:

KtoNr	**Saldo**
1	100
2	100

```
-- begin transaction isolation level serializable;
-- begin transaction isolation level read committed;
select Saldo from Konto where KtoNr = 1;
-- hier funkt die 2. Transaktion dazwischen
if (Saldo > 60) {
  update Konto set Saldo = Saldo - 60 where KtoNr = 1;
  update Konto set Saldo = Saldo + 60 where KtoNr = 2;
}
commit;
```

Was passiert bei den beiden Isolationsleveln??