

- Anforderungen an Transaktionen
- Mechanismen
- Isolationsstufen
- Snapshot Isolation
- Anomalien
- Postgres - Snapshot Isolation
- Postgres- Beispiele für Anomalien
- Postgres - weiter Beispiele

Atomarität (Atomicity)

- Alles oder Nichts
- Alle Aktionen einer Transaktion werden bestätigt oder keine

Konsistenz (Consistency)

- Eine Transaktion kann nur bestätigt werden, wenn keine Konsistenbedingungen verletzt sind
- Ist eigentlich ein Konzept der Anwendung

Isolation (Isolation)

- Nebenläufige/Parallele Transaktionen beeinflussen sich nicht auf unerlaubte Art und Weise.
- Es gibt verschiedene Stufen der Isolation, deren stärkste Serialisierbarkeit ist
- Stärkere Stufe bedeutet geringere Leistung

Dauerhaftigkeit (Durability)

- Veränderungen bestätigter Transaktionen haben für Folgetransaktionen Bestand
- Solange bis sie durch weitere Transaktionen bestätigt verändert werden
- Auch im Falle von Fehlern (z.B. Festplatte defekt)

Strikte 2-Phasen-Sperren (Strict two-phase locking , 2PL)

- Lesesperren (Shared Locks , Slocks) vor jedem Lesen
- Schreibsperren (Exclusive Locks, Xlocks) vor jedem Schreiben
- Sperren auf existierende Datensätze (Row Level Locks)
- Sperren basierend auf Prädikate (Predicate Locks) zur Verhinderung von Phantomen
- Sperren werden bis zum Transaktionsende gehalten

Optimistic Concurrency Control (OCC)

- Lesen und Schreiben ohne Sperren (auf Kopien)
- Führen der Historie der Lese- und Schreibvorgänge
- Überprüfung der Historie auf Konflikte vor Commit
- Abbruch bei Konflikt

Multi-Version Concurrency Control (MVCC) / Snapshot Isolation

- Keine Lesesperren
- Änderungen erzeugen neue Versionen
- Transaktionen sehen Version der letzten bestätigten Änderung, d.h. Lesen und Schreiben stehen nicht in Konflikt
- Prüfung der Schreibvorgänge, Abbruch bei Konflikt

Optimistische Verfahren auf Anwendungsebene

- Explizite Versionsfelder
- Änderung nur, wenn keine Versionsänderung in Datenbank
- Bsp. OR-Mapper

SQL-Isolationsstufen

Read Uncommitted

- Lesen nichtbestätigter Änderungen möglich

Read Committed

- Lesen nur von bestätigten Änderungen möglich

Repeatable Read

- Wiederholtes Lesen in der Transaktion liefert gleichen Wert

Serializable

- Nebenläufige Ausführung von Transaktionen muss einer serialisierten Ausführung entsprechen

Produktspezifische Isolationsstufen

Cursor Stability (DB2, ..)

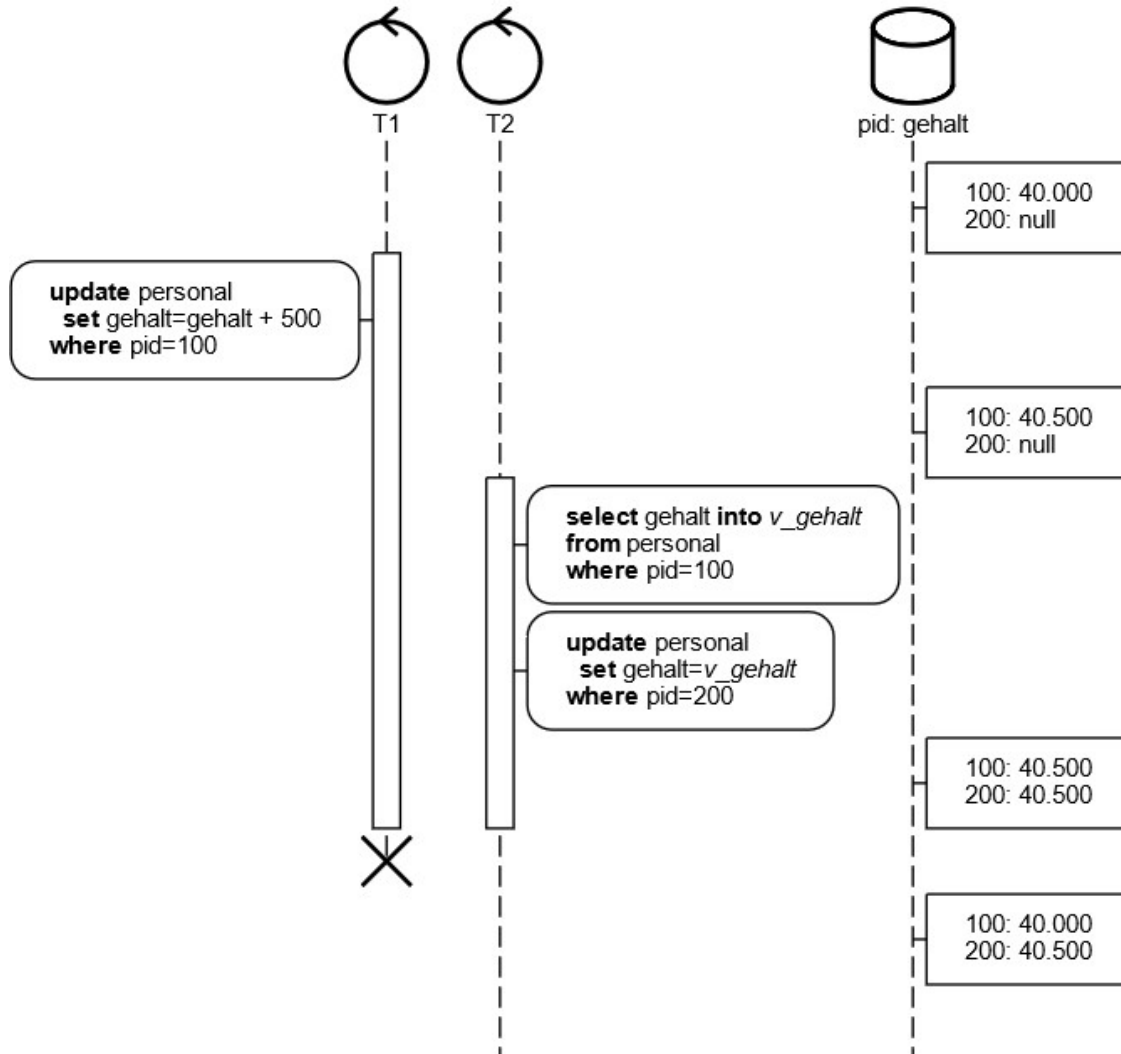
- Basiert auf Datenbank-Cursor
- Sperre bleibt solange bestehen, bis auf den nächsten Datensatz im Cursor weitergeschaltet wird

Snapshot Isolation (PostgreSQL, ...)

- First Committer wins

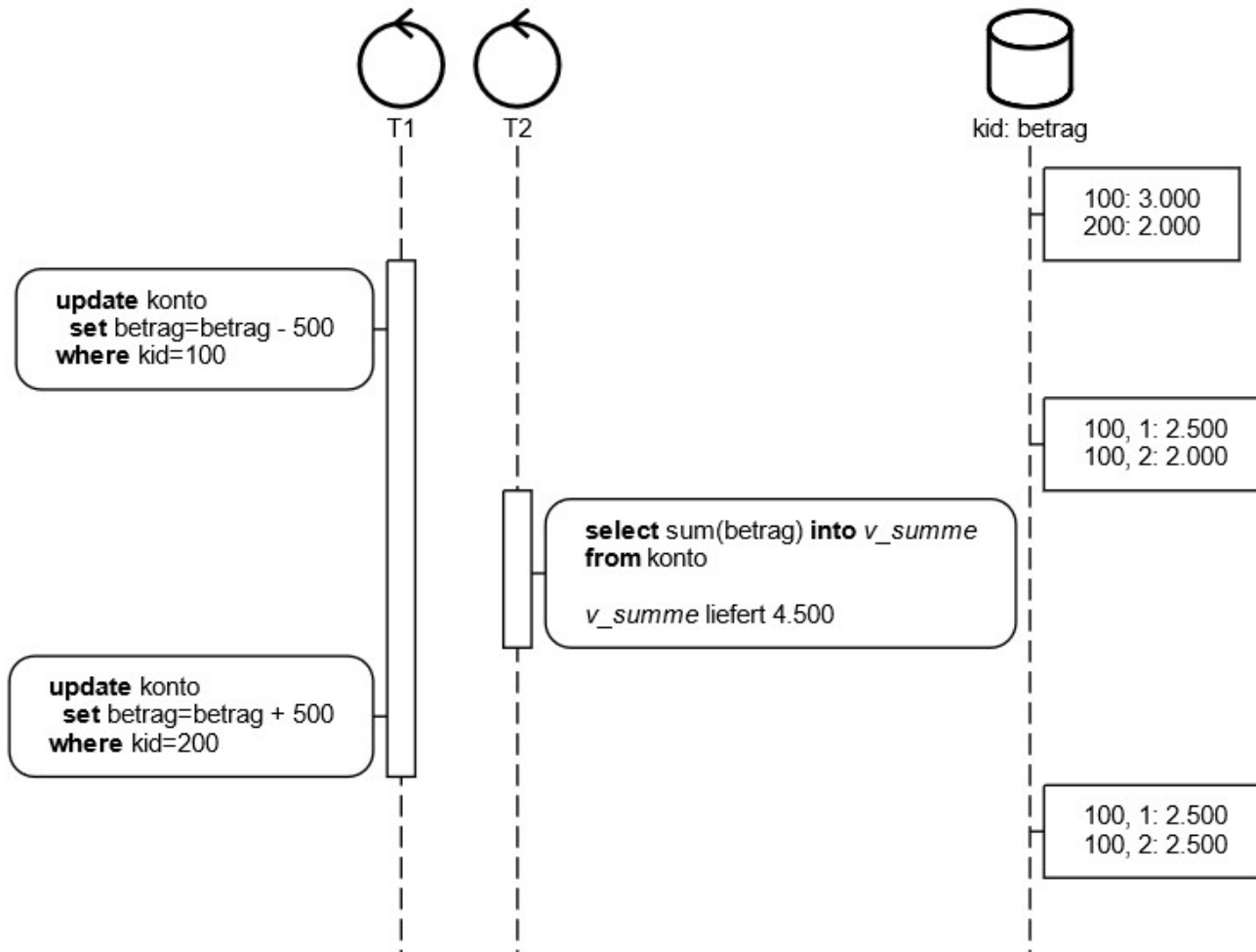
Read Consistency (Oracle)

- First Updater wins

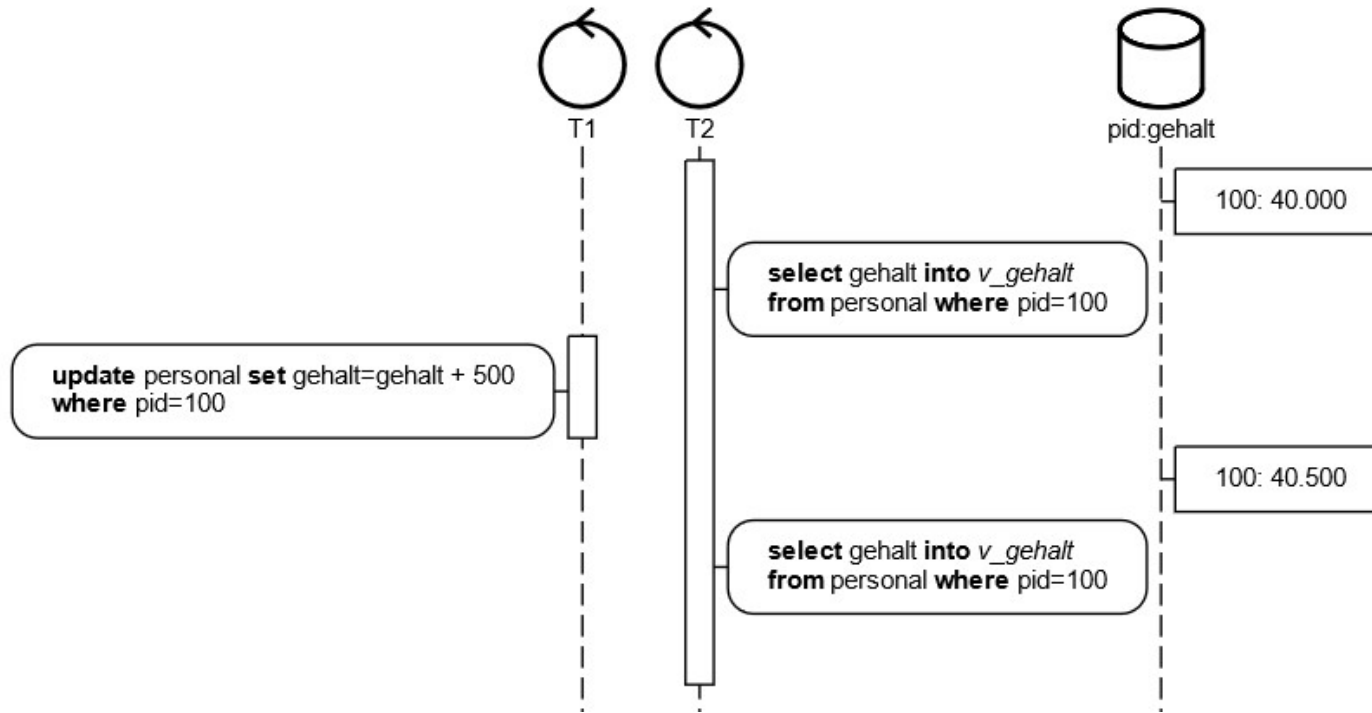


Mitarbeiterin mit pid=200 soll dasselbe Gehalt haben wie die mit pid=100

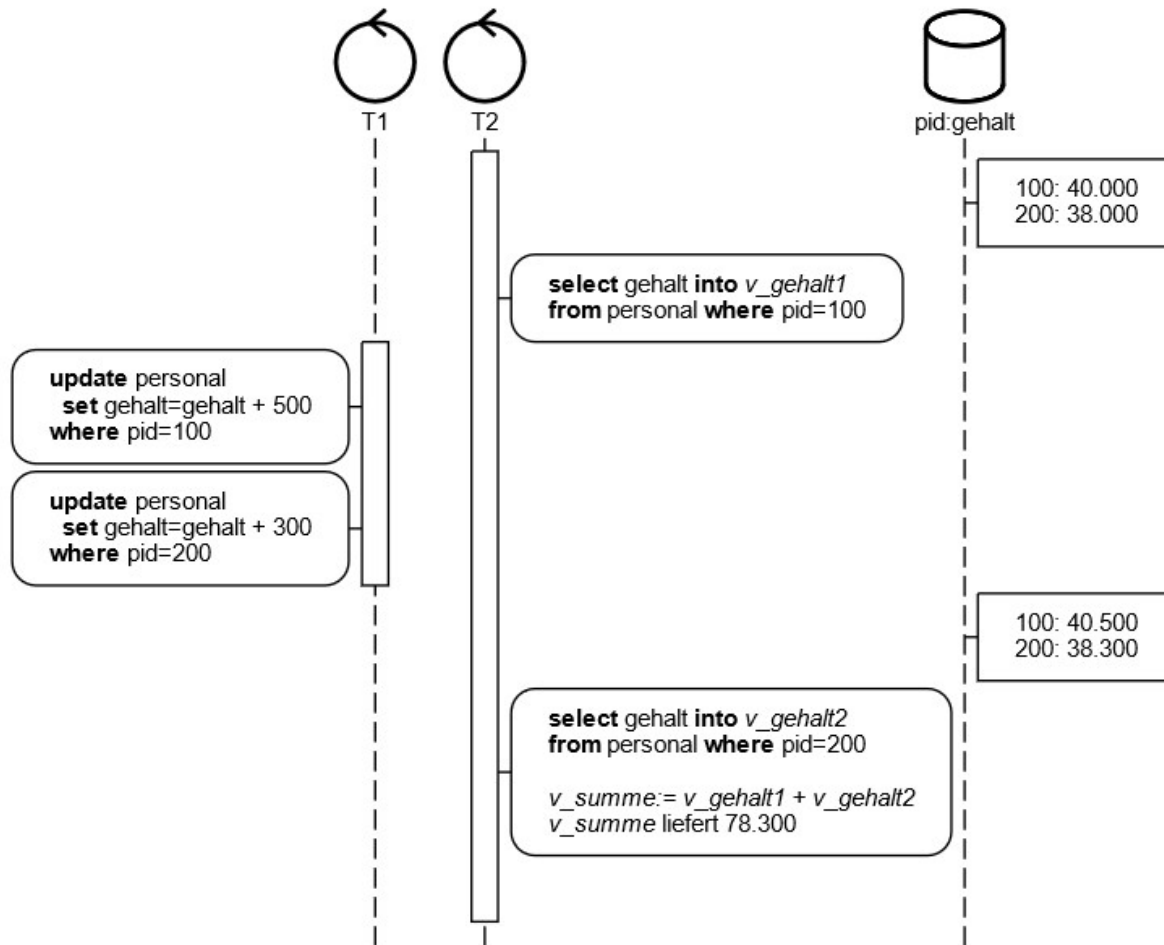
Lesen eines nicht bestätigten Datenbankzustands



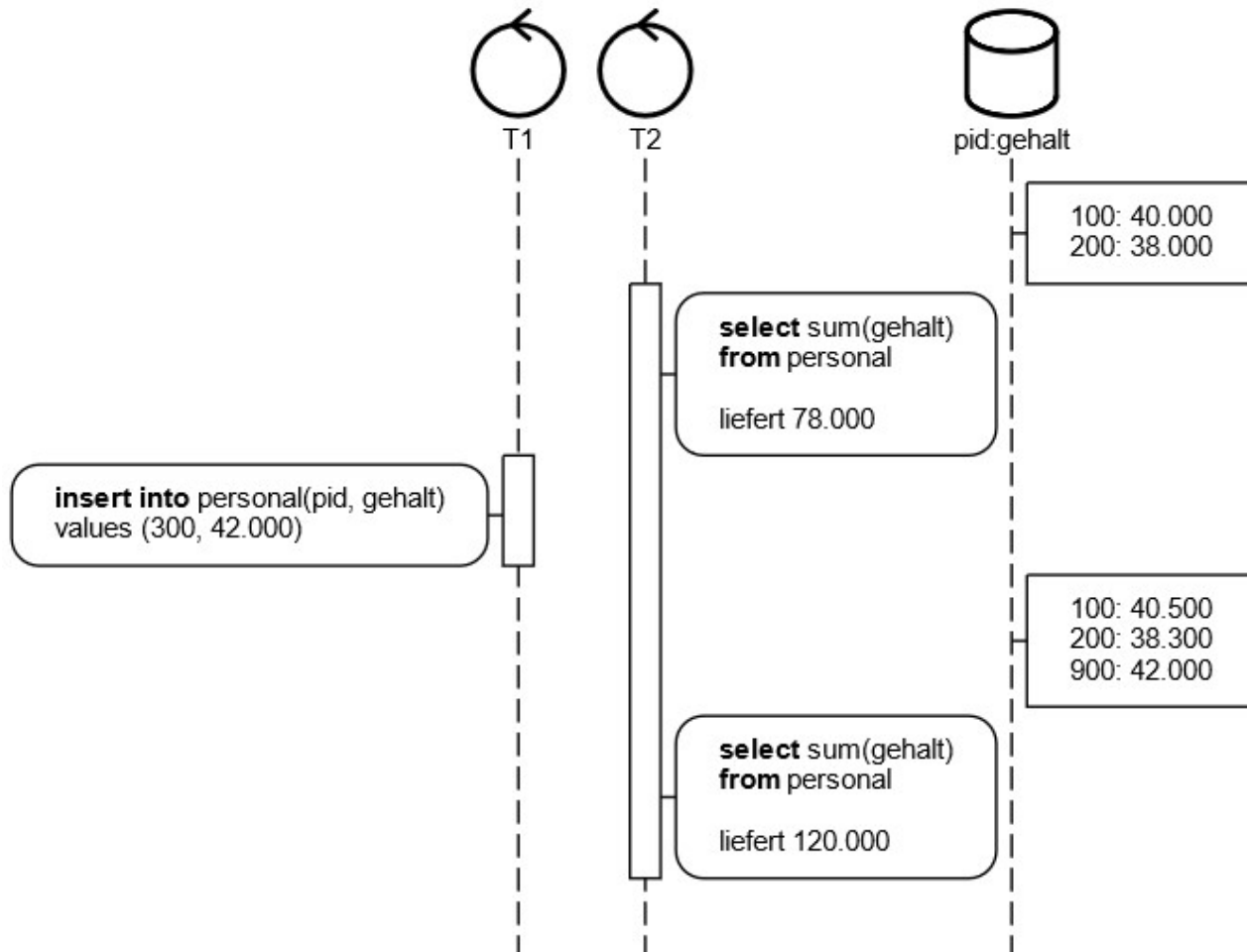
Lesen eines nicht bestätigten
Datenbankzustands
Ähnlich wie Dirty Read, nur kein
Rollback



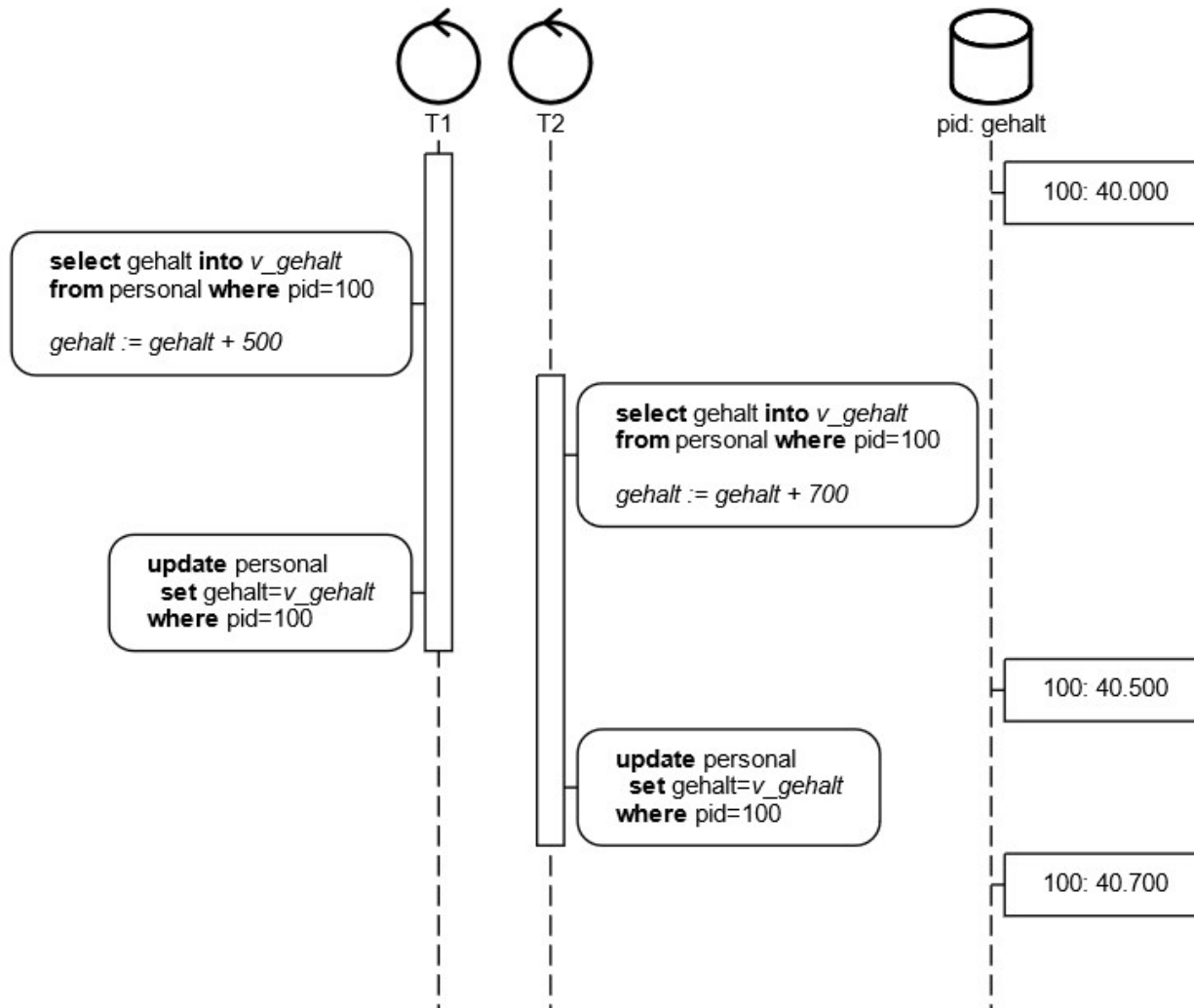
Wiederholtes Lesen des gleichen Datensatzes innerhalb einer Transktion führt zu unterschiedlichen Werten



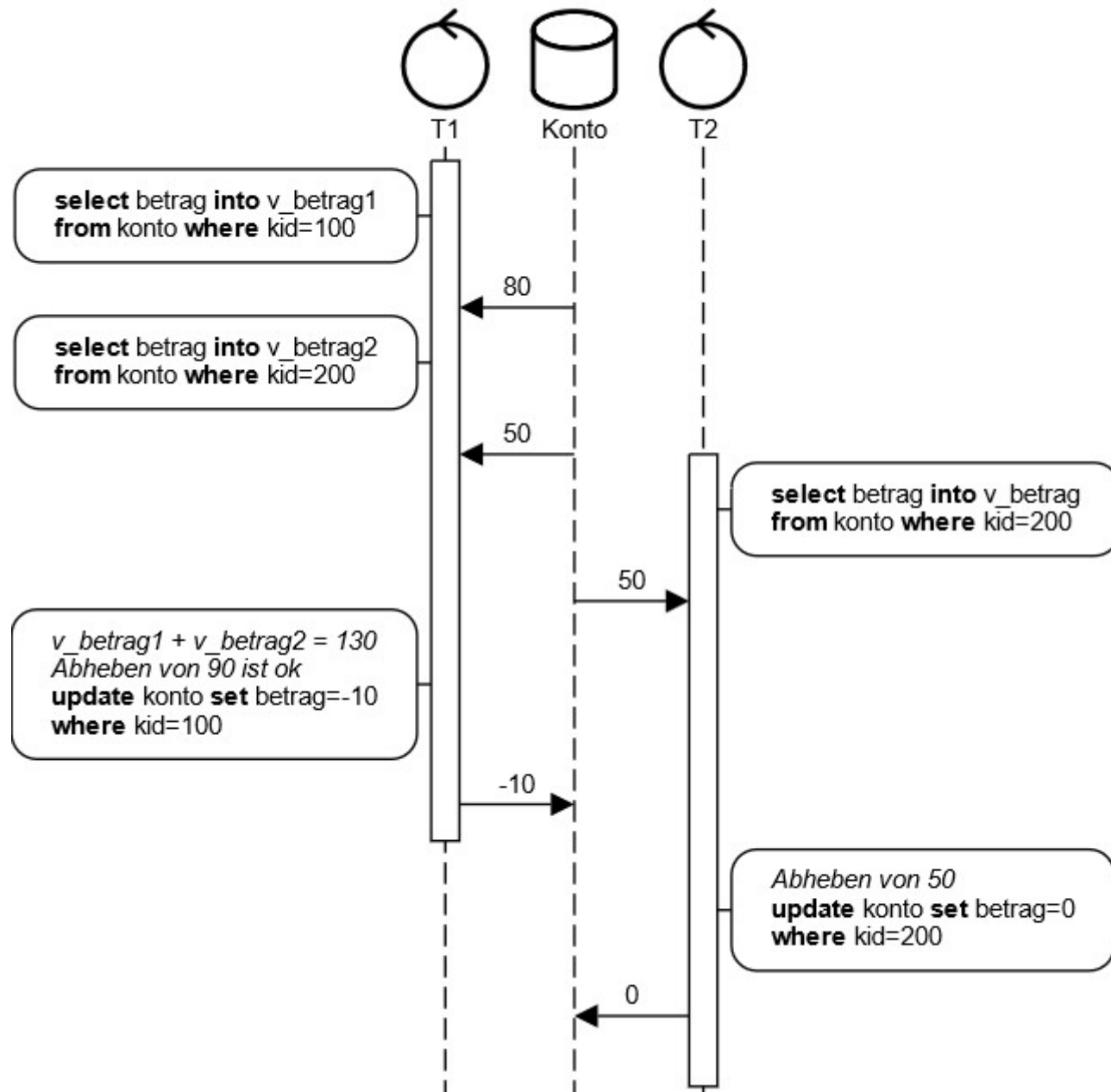
Ähnlich zu Non Repeatable Read
Nur nicht zweimaliges Lesen des gleichen Datensatzes
Der Gesamtzustand der Datenbank hat sich beim zweiten Lesen geändert



Ähnlich zu Read Skew 2
Der Gesamtzustand der Datenbank hat sich
beim zweiten Lesen geändert



Unerlaubtes Überschreiben einer Änderung



Invariante: Summe auf beiden Konten darf nicht negativ werden

T1 prüft die Summe (130) und stellt fest, dass ein Abheben von 90 von kid=100 ok ist, da Deckung durch kid=200 gegeben

T2 hebt ebenfalls Geld ab

Dadurch wird die Invariante verletzt

konto	kid betrag	T1	T2	T3	T
	<pre> -----+ 100 2 200 6 </pre>				
T1	<pre> begin transaction isolation level serializable; -- T1 update konto set betrag = 3 where kid=100; select * from konto order by kid; commit; </pre>	<pre> kid betrag -----+ 100 3 200 6 </pre>			
T	<pre> begin transaction isolation level serializable; -- T </pre>				
T2	<pre> begin transaction isolation level serializable; -- T2 update konto set betrag = 7 where kid=200; select * from konto order by kid; commit; </pre>		<pre> kid betrag -----+ 100 3 200 7 </pre>		
T3	<pre> begin transaction isolation level serializable; -- T3 insert into konto values (300, 1); select * from konto order by kid; commit; </pre>			<pre> kid betrag -----+ 100 3 200 7 300 1 </pre>	
T	<pre> select * from konto order by kid; commit; </pre>				<pre> kid betrag -----+ 100 3 200 6 </pre>

konto	kid betrag ----+-----+ 100 2 200 6	T1	T2	T3	T
T1	<code>begin transaction isolation level serializable; -- T1</code> <code>update konto set betrag = 3 where kid=100;</code> <code>select * from konto order by kid;</code> <code>commit;</code>	kid betrag ----+-----+ 100 3 200 6			
T	<code>begin transaction isolation level read committed; -- T</code>				
T2	<code>begin transaction isolation level serializable; -- T2</code> <code>update konto set betrag = 7 where kid=200;</code> <code>select * from konto order by kid;</code> <code>commit;</code>		kid betrag ----+-----+ 100 3 200 7		
T3	<code>begin transaction isolation level serializable; -- T3</code> <code>insert into konto values (300, 1);</code> <code>select * from konto order by kid;</code> <code>commit;</code>			kid betrag ----+-----+ 100 3 200 7 300 1	
T	<code>select * from konto order by kid;</code> <code>commit;</code>				kid betrag ----+-----+ 100 3 200 7 300 1

konto	kid betrag	T1	T2	T3	T
	<pre> -----+ 100 2 200 6 </pre>				
T1	<pre> begin transaction isolation level serializable; -- T1 update konto set betrag = 3 where kid=100; select * from konto order by kid; commit; </pre>	<pre> kid betrag -----+ 100 3 200 6 </pre>			
T	<pre> begin transaction isolation level read committed; -- T </pre>				
T2	<pre> begin transaction isolation level serializable; -- T2 update konto set betrag = 7 where kid=200; select * from konto order by kid; </pre>		<pre> kid betrag -----+ 100 3 200 7 </pre>		
T3	<pre> begin transaction isolation level serializable; -- T3 insert into konto values (300, 1); select * from konto order by kid; </pre>			<pre> kid betrag -----+ 100 3 200 6 300 1 </pre>	
T	<pre> select * from konto order by kid; commit; </pre>				<pre> kid betrag -----+ 100 3 200 6 </pre>
T2	<pre> commit; </pre>				
T3	<pre> commit; </pre>			Serialisierungsfehler	
T	<pre> select * from konto order by kid; </pre>				<pre> kid betrag -----+ 100 3 200 7 </pre>

konto	kid betrag	T1	T2	T3	T
	<pre> -----+ 100 2 200 6 </pre>				
T1	<pre> begin transaction isolation level serializable; -- T1 update konto set betrag = 3 where kid=100; select * from konto order by kid; commit; </pre>	<pre> kid betrag -----+ 100 3 200 6 </pre>			
T	<pre> begin transaction isolation level read committed; -- T </pre>				
T2	<pre> begin transaction isolation level serializable; -- T2 update konto set betrag = 7 where kid=200; select * from konto order by kid; </pre>		<pre> kid betrag -----+ 100 3 200 7 </pre>		
T3	<pre> begin transaction isolation level serializable; -- T3 insert into konto values (300, 1); select * from konto order by kid; (wird nicht ausgeführt) </pre>				
T	<pre> select * from konto order by kid; commit; </pre>				<pre> kid betrag -----+ 100 3 200 6 </pre>
T2	<pre> commit; </pre>				
T3	<pre> commit; </pre>				
T	<pre> select * from konto order by kid; </pre>				<pre> kid betrag -----+ 100 3 200 7 300 1 </pre>

personal	pid gehalt ----+-----+ 100 40000 200 38000	T1	T2
T2	<code>begin transaction isolation level read committed; -- T2</code> <code>select * from personal where pid=100;</code>		pid gehalt ----+-----+ 100 40000
T1	<code>begin transaction isolation level read committed; -- T1</code> <code>update personal set gehalt=gehalt + 500 where pid=100;</code> <code>update personal set gehalt=gehalt + 300 where pid=200;</code> <code>select * from personal order by pid;</code> <code>commit;</code>	pid gehalt ----+-----+ 100 40500 200 38300	
T2	<code>select * from personal where pid=200;</code> <code>commit;</code>		pid gehalt ----+-----+ 200 38300

personal	pid gehalt ----+-----+ 100 40000 200 38000	T1	T2
T2	<code>begin transaction isolation level repeatable read; -- T2</code> <code>select * from personal where pid=100;</code>		pid gehalt ----+-----+ 100 40000
T1	<code>begin transaction isolation level read committed; -- T1</code> <code>update personal set gehalt=gehalt + 500 where pid=100;</code> <code>update personal set gehalt=gehalt + 300 where pid=200;</code> <code>select * from personal order by pid;</code> <code>commit;</code>	pid gehalt ----+-----+ 100 40500 200 38300	
T2	<code>select * from personal where pid=200;</code> <code>commit;</code>		pid gehalt ----+-----+ 200 38000

personal	pid gehalt ----+-----+	T1	T2	T
	100 40000			
T1	<code>begin transaction isolation level read committed; -- T1</code> <code>select * from personal where pid=100;</code>	pid gehalt ----+-----+		
T2	<code>begin transaction isolation level read committed; -- T2</code> <code>select * from personal where pid=100;</code>		pid gehalt ----+-----+	
T1	<i>-- erhöhe gehalt auf 40500</i> <code>update personal set gehalt=40500 where pid=100;</code> <code>select * from personal where pid=100;</code> <code>commit;</code>	pid gehalt ----+-----+		
T2	<i>-- erhöhe gehalt auf 40700</i> <code>update personal set gehalt=40700 where pid=100;</code> <code>select * from personal where pid=100;</code> <code>commit;</code>		pid gehalt ----+-----+	
T	<code>select * from personal where pid=100;</code>			pid gehalt ----+-----+
				100 40700

personal	pid gehalt ----+-----+ 100 40000	T1	T2
T1	<code>begin transaction isolation level serializable; -- T1</code> <code>select * from personal where pid=100;</code>	pid gehalt ----+-----+ 100 40000	
T2	<code>begin transaction isolation level serializable; -- T2</code> <code>select * from personal where pid=100;</code>		pid gehalt ----+-----+ 100 40000
T1	<i>-- erhöhe gehalt auf 40500</i> <code>update personal set gehalt=40500 where pid=100;</code> <code>select * from personal where pid=100;</code> <code>commit;</code>	pid gehalt ----+-----+ 100 40500	
T2	<i>-- erhöhe gehalt auf 40700</i> <code>update personal set gehalt=40700 where pid=100;</code> <code>select * from personal where pid=100;</code> <code>commit;</code>		Serialisierungsfehler
T	<code>select * from personal where pid=100;</code>		pid gehalt ----+-----+ 100 40500

konto	kid betrag -----+ 100 80 200 50	T1	T2	T
T1	<code>begin transaction isolation level read committed; -- T1</code> <code>select * from konto where kid=100;</code>	kid betrag -----+ 100 80		
T1	<code>select * from konto where kid=200;</code>	kid betrag -----+ 200 50		
T2	<code>begin transaction isolation level read committed; -- T2</code> <code>select * from konto where kid=200;</code>		kid betrag -----+ 200 50	
T1	<code>update konto set betrag = -10 where kid=100;</code> <code>select * from konto where kid=100;</code> <code>commit;</code>	kid betrag -----+ 100 -10		
T2	<code>update konto set betrag = 0 where kid=200;</code> <code>select * from konto where kid=200;</code> <code>commit;</code>		kid betrag -----+ 200 0	
T	<code>select * from konto order by kid;</code>			kid betrag -----+ 100 -10 200 0

konto	kid betrag	T1	T2
	<pre> -----+ 100 80 200 50 </pre>		
T1	<pre> begin transaction isolation level serializable; -- T1 select * from konto where kid=100; </pre>	<pre> kid betrag -----+ 100 80 </pre>	
T1	<pre> select * from konto where kid=200; </pre>	<pre> kid betrag -----+ 200 50 </pre>	
T2	<pre> begin transaction isolation level serializable; -- T2 select * from konto where kid=200; </pre>		<pre> kid betrag -----+ 200 50 </pre>
T1	<pre> update konto set betrag = -10 where kid=100; select * from konto where kid=100; commit; </pre>	<pre> kid betrag -----+ 100 -10 </pre>	
T2	<pre> update konto set betrag = 0 where kid=200; select * from konto where kid=200; </pre>		Serialisierungsfehler
T2	<pre> commit; </pre>		
T	<pre> select * from konto order by kid; </pre>		<pre> kid betrag -----+ 100 -10 200 50 </pre>

Unterschiedliche Snapshots bei Read Committed und Repeatable Read

kv	k v	T1	T2	T3
	<pre> ---+-----+ 10 10-a </pre>			
T2	<code>begin transaction isolation level read committed; -- T2</code>			
T3	<code>begin transaction isolation level repeatable read; --T3</code>			
T1	<pre> begin transaction isolation level read committed; -- T1 update kv set v = '10-b' where k=10; select * from kv; commit; </pre>	<pre> k v ---+-----+ 10 10-b </pre>		
T2	<pre> select * from kv; commit; </pre>		<pre> k v ---+-----+ 10 10-b </pre>	
T3	<pre> select * from kv; commit; </pre>			<pre> k v ---+-----+ 10 10-a </pre>

kv	k v ---+-----+ 10 10-a	T1	T2	T
T1	<code>begin transaction isolation level serializable; -- T1</code> <code>update kv set v = '10-b' where k=10;</code> <code>select * from kv;</code>	k v ---+-----+ 10 10-b		
T2	<code>begin transaction isolation level serializable; -- T2</code> <code>update kv set v = '10-c' where k=10;</code>		wartet	
T1	<code>commit;</code>		Serialisierungsfehler	
T2	<code>commit;</code>			
T	<code>select * from kv;</code>			k v ---+-----+ 10 10-b

kv	k v ---+-----+ 10 10-a	T1	T2	T
T1	<code>begin transaction isolation level read committed; -- T1</code> <code>update kv set v = '10-b' where k=10;</code> <code>select * from kv;</code>	k v ---+-----+ 10 10-b		
T2	<code>begin transaction isolation level read committed; -- T2</code> <code>update kv set v = '10-c' where k=10;</code>		wartet	
T1	<code>commit;</code>		läuft weiter	
T2	<code>commit;</code>			
T	<code>select * from kv;</code>			k v ---+-----+ 10 10-c