

Datenbanken und Informationssysteme

Isolationslevel und Synchronisationskontrolle

Burkhardt Renz

Fachbereich MNI
TH Mittelhessen

Sommersemester 2020

Übersicht

- Isolationslevel
 - Phänomene
 - Definition der Isolationslevel
- Implementierung von Isolationsleveln
 - Sperrmechanismen
 - Multiversionierung
- Verwendung von Isolationsleveln
 - Konzeptionelle Überlegungen
 - Beispiele

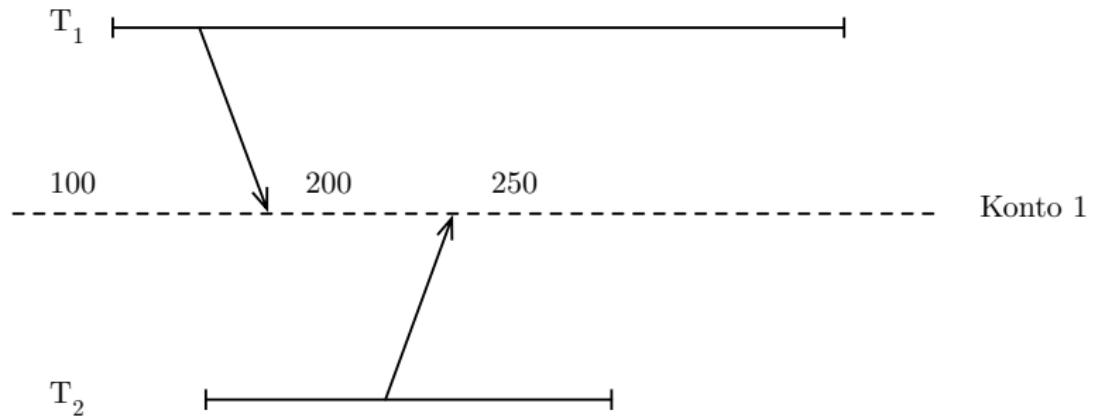
Lost Update[1]/Dirty Write

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

T1 geht von einem Saldo von 200 aus



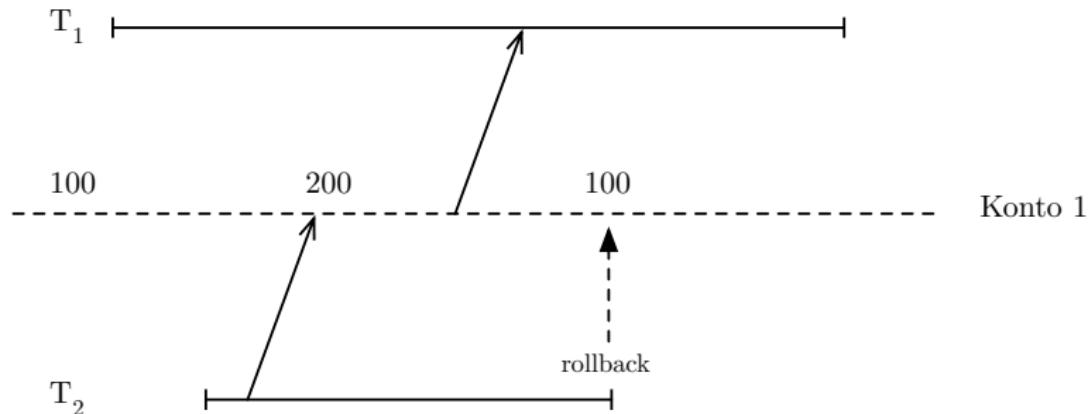
Dirty Read

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
		100
commit		

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

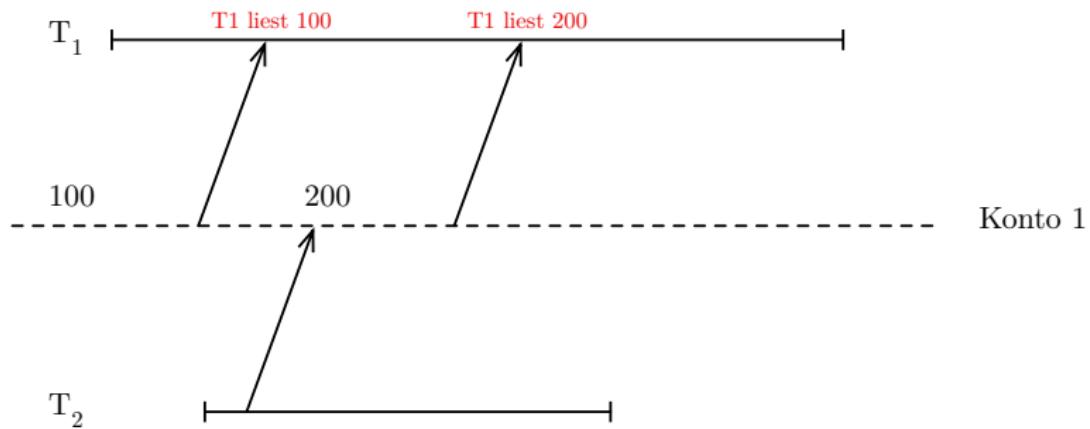
```
select Saldo  
from Konto  
where KtoNr = 1  
 $T_1$  liest den Wert 100
```

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.

Lost Update[2]

Folgendes Phänomen ist bei Non-repeatable Read möglich:

T_1	T_2	Saldo für Konto 1
		100

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

```
select Saldo  
from Konto  
where KtoNr = 1  
 $T_2$  liest den Wert 100
```

```
update Konto  
set Saldo = 100+100  
where KtoNr = 1  
...  
commit
```

200

```
update Konto  
set Saldo = 100+50  
where KtoNr = 1  
...  
commit
```

150

Phantom Row

T_1	T_2	Saldo für Konten
		100, 100

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

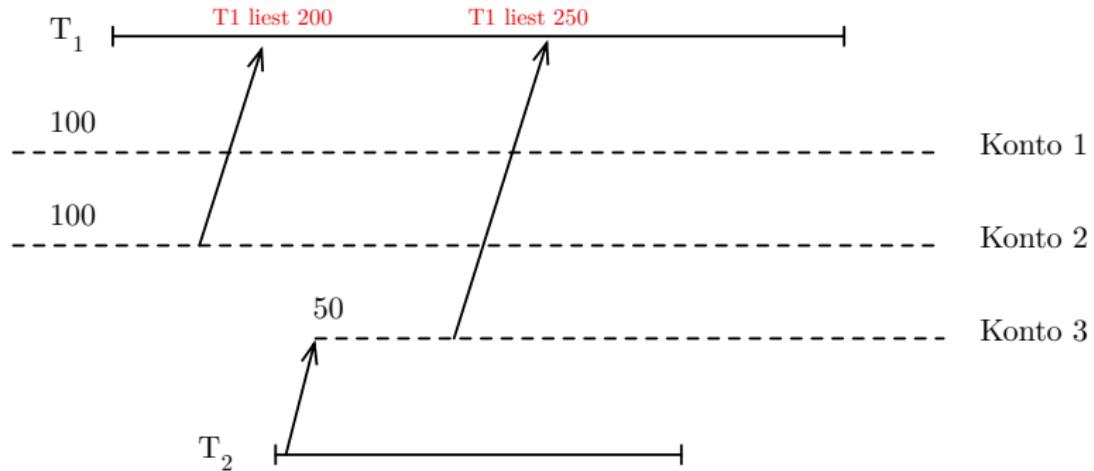
```
insert into Konto
values (3,50)
commit
```

100, 100, 50

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

Transaktion T_1 bekommt Datensätze „untergeschoben“!

Phantom Row



Definition der Isolationslevel in SQL

	Dirty Read	Non-Repeable Read	Phantom Row
READ UNCOMMITTED	möglich	möglich	möglich
READ COMMITTED	<i>nicht</i> möglich	möglich	möglich
REPEATABLE READ	<i>nicht</i> möglich	<i>nicht</i> möglich	möglich
SERIALIZABLE	<i>nicht</i> möglich	<i>nicht</i> möglich	<i>nicht</i> möglich

Übersicht

- Isolationslevel
 - Phänomene
 - Definition der Isolationslevel
- Implementierung von Isolationsleveln
 - Sperrmechanismen
 - Multiversionierung
- Verwendung von Isolationsleveln
 - Konzeptionelle Überlegungen
 - Beispiele

Konzepte der Implementierung von Isolationsleveln

- Sperrmechanismen (DB2)
- Multiversionierung (Oracle)

Diskussion der Konzepte - siehe Skript auf meiner Webseite

Modussperren und Prädikatsperren

Sperrverfahren

Bei Sperrverfahren werden Datenobjekte mit Sperren (*locks*) versehen, die den Zugriff von Transaktionen auf diese Datenobjekte einschränken.

Arten von Sperren

- Lesesperre, auch *shared lock*
- Schreibsperre, auch *exclusive lock*
- Prädikatsperren

Dauer von Sperren

Dauer von Sperren

- Kurze Sperren – während des eigentlichen Zugriffs auf ein Datenobjekt
- Lange Sperren – gehalten bis zum Ende der Transaktion

Vorgehen mit Sperren

- **READ UNCOMMITTED:**

lesend: berücksichtigt keine Sperren
schreibend: in SQL nicht erlaubt

- **READ COMMITTED:**

lesend: kurze Lese-Sperren
schreibend: lange exklusive Prädikatsperren

- **REPEATABLE READ:**

lesend: lange Lese-Sperren
schreibend: lange exklusive Prädikatsperren

- **SERIALIZABLE:**

lesend: lange nicht-exklusive Prädikatsperren
schreibend: lange exklusive Prädikatsperren

Multiversionierung

- Es kann verschiedene Versionen eines Datenobjekts geben.
- Globale Versionsnummer gibt die aktuellste Version an.
- Bei jedem schreibenden Zugriff wird die globale Versionsnummer erhöht und das geschriebene Datenobjekt hat diese Versionsnummer.
- Eine Transaktion kann einen Schnapschuss (*snapshot*) lesen: sie sieht diejenigen Datenobjekte, deren Version am nächsten unterhalb der globalen Versionsnummer zu Beginn der Transaktion sind.

Read-Only Multiversion Concurrency Control

- Transaktion gibt zu Beginn an, ob sie nur lesend ist, oder auch schreiben möchte.
- READ ONLY: erhält einen Schnappschuss der Datenbank zum Zeitpunkt des ersten Zugriffs
- schreibend: striktes 2-Phasen-Lock-Protokoll
- Eigenschaften:
 - Lesende Transaktionen müssen niemals auf schreibende Transaktionen warten
 - Schreibende Transaktionen müssen niemals auf lesende Transaktionen warten

Read-Consistency Multiversion Concurrency Control

- Transaktion gibt zu Beginn an, ob sie nur lesend ist, oder auch schreiben möchte.
- READ ONLY: erhält einen Schnappschuss der Datenbank zum Zeitpunkt des ersten Zugriffs
- Schreibende Aktionen innerhalb einer Transaktion verwenden lange Schreibsperrungen
Lesende Aktionen erhalten stets die aktuellste Version eines Datenobjekts
- Eigenschaften: Lesende Transaktionen müssen niemals auf schreibende Transaktionen warten
Schreibende Transaktionen müssen niemals auf lesende Transaktionen warten
- typischerweise die Implementierung von READ COMMITTED

Snapshot Isolation

- Keine vorherige Unterscheidung lesend/schreibend
- Lesende Zugriffe erhalten Werte des Schnapschusses
- Schreibende Zugriffe müssen die *disjoint write property* erfüllen: verschiedene Transaktionen dürfen niemals dasselbe Datenobjekt ändern.
- Wird diese Eigenschaft von einer Transaktion verletzt, wird sie abgebrochen.
- typischerweise die Implementierung von SERIALIZABLE

Szenario

- Ein Kunde hat zwei Konten bei einer Bank.
- Die Bank hat mit ihm vereinbart, dass er zwar eines der Konten überziehen darf, aber nur, wenn der Betrag durch das andere Konto gedeckt ist.
D.h. die Summe der Salden beider Konten muss stets ≥ 0 sein.
- Was kann bei *Snapshot Isolation* passieren?

Write Skew

T_1	T_2	Salden der Konten
		1: 100 2: 100
select Saldo ... where KtoNr in (1,2) T_1 liest die Werte 100,100	select Saldo ... where KtoNr in (1,2) T_2 liest den Wert 100,100	
update Konto set Saldo = 100-120 where KtoNr = 1 ... commit		Konto 1:-20
	update Konto set Saldo = 100-120 where KtoNr = 2 ... commit	Konto 2: -20

Übersicht

- Isolationslevel
 - Phänomene
 - Definition der Isolationslevel
- Implementierung von Isolationsleveln
 - Sperrmechanismen
 - Multiversionierung
- Verwendung von Isolationsleveln
 - Konzeptionelle Überlegungen
 - Beispiele

Motivation

Beispiel aus einer Anwendung

```
try {  
    con.setTransactionIsolation(  
        Connection.TRANSACTION_SERIALIZABLE);  
    // tue etwas  
    con.commit();  
} (catch e) {  
    JOptionPane.showMessageDialog("Fehler");  
    con.rollback();  
}
```

Worin besteht das Problem?

Wie kann man es beheben?

Datenbank- und Geschäftstransaktionen

Arten von Transaktionen

Geschäftstransaktion

Datenbanktransaktion

Arten von Datenbanktransaktionen

Kurze Transaktion

Lange Transaktion

Strategien

Optimistische Strategie

Die Geschäftstransaktion wird in mehrere kurze Datenbanktransaktionen zerlegt.

Beim Commit der kurzen Transaktionen wird überprüft, ob die Konsistenz der Daten gewahrt bleibt.

Prüfung kann in Layer (z.B. JPA) oder durch die Anwendung durchgeführt werden.

Pessimistische Strategie

Die Geschäftstransaktion wird in einer langen Datenbanktransaktion durchgeführt.

Deshalb kann die Anwendung sicher sein, dass keine andere Transaktion ihre Daten beeinflussen kann (je nach Isolationslevel). Oft durch explizite anwendungsseitige Sperren realisiert (z.B. JPA)

Coyote Buttes South



Alte Oper Frankfurt



Jeff Beck



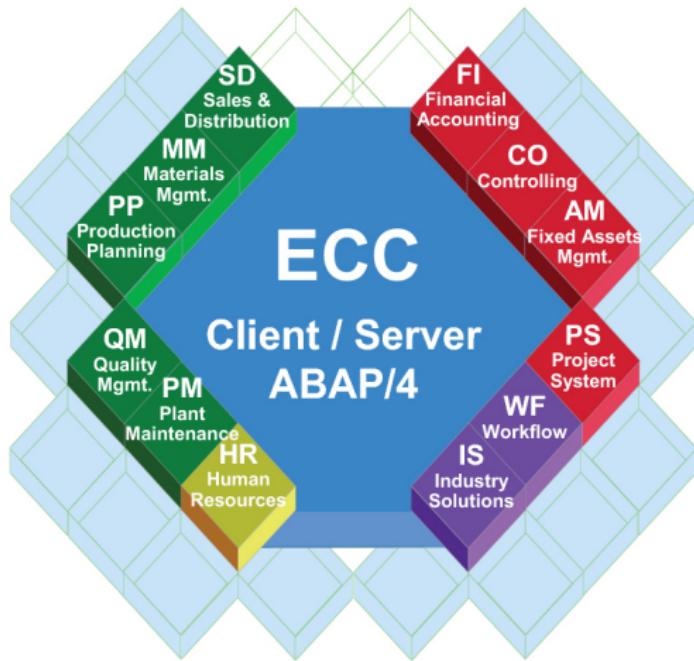
Fela!



Grundbuch

16.04.2010: Becker				Anlage
Erste Abteilung (Spalten 1 bis 4)				Freimersheim -Rheinhessen- 12
LNrE	Eigentümer	LNrG	Grundlage der Eintragung	
1.1	[REDACTED]	1	Auflassung vom 15.01.2010; eingetragen am 16.04.2010.	
1.2	zu 1/2-Anteil [REDACTED]			
			zu 1/2-Anteil [REDACTED]	
Zweite Abteilung (Spalten 1 bis 3)				Freimersheim -Rheinhessen- 12
LNrE	LNrG	Lasten und Beschränkungen		
1	1	Vormerkung zur Sicherung des bedingten Anspruchs auf Rückübertragung für [REDACTED] Rang nach Abt. III Nr. 1. Gemäß Bewilligung vom 15.01.2010 ([REDACTED]) eingetragen am 16.04.2010. [REDACTED]		
Dritte Abteilung (Spalten 1 bis 4)				Freimersheim -Rheinhessen- 12
LNrE	LNrG	Betrag	Hypothesen, Grundschulden, Rentenschulden	
1	1	195.000 EUR	Grundschatz ohne Brief zu einhundertfünfundneunzigtausend Euro für [REDACTED] 12 % Zinsen jährlich. Vollstreckbar nach § 800 ZPO. Rang vor Abt. II Nr. 1. Gemäß Bewilligung vom 15.01.2010 ([REDACTED]) eingetragen am 16.04.2010.	

Bestellprozess



Oper Frankfurt



©2010 N. Gasteiger