JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Isolationsstufen und Sperren

Die nebenläufige Verwendung von Datensätzen in Datenbanken

Informationstechnische Systeme

4AHITM 2015/16

Benedikt Berger & Antonio Pavic

Version 1.0

Note:

Begonnen am 2. Dezember 2015

Betreuer: Prof. Borko

Beendet am 9. Dezember 2015

Inhaltsverzeichnis

1. Angabe

Ziele

Diese Übung soll Einblick in die nebenläufige Verwendung von Datensätzen in Datenbanken gewähren.

Voraussetzungen

Kenntnis über Transaktionskonzepte (ACID) sowie der theoretische Background zu Isolationslevel und Sperren.

Aufgabenstellung

Zeige die möglichen Fehlerfälle bei Transaktionen an enstprechenden Beispielen. Es soll klar ersichtlich sein, in welcher Isolationsstufe welche Fehler auftreten können.
Verwende dafür entweder Screenshots oder klar strukturierte Tabellen. Es ist aber notwendig diese in den jeweiligen DBMS zu testen. Zeige auch den Fehlerfall des Lost-Updates!

Dokumentiere dabei alle notwendigen Schritte zur Verwendung von Transaktionen (autocommit, tx_isolation, etc.)

Verwende explizite Sperren und setze diese in Vergleich zu den Isolationslevel bzw. zeige wie man mit Locks die aufgetreten Fehlerfälle vermeiden kann.
Gebe alles in einem PDF ab. Vergiss dabei nicht auf die Anwendung der Zitierregeln. Bitte darauf zu achten, dass kein schwarzer Toner bei einem möglichen Ausdruck verschwendet wird. Es ist eine Gruppengröße von zwei Schülern erlaubt!

2. ACID

Das **ACID-Prinzip** beschreibt gewünschte Eigenschaften von Transaktionen in einer Datenbank. Eine Datenbanktransaktion ist eine Sequenz von Operationen, die einzigartig und durchführbar sein muss.

Atomicity:

Eine Datei-Operation muss entweder ganz oder garnicht ausgeführt werden.

Consistancy:

Man spricht von einer vorhandenen Datenkonsistenz, wenn nach einer Sequenz der Datenzustand in einem konsistenten Zustand zurückgelassen wird.

Isolation:

Die Isolation verhindert, dass sich parallele Abläufe auf Datei-Operationen gegenseitig beeinflussen können.

Durability:

Die Dauerhaftigkeit versichert eine dauerhafte Speicherung der Datei-Operationen auf einem Datenträger.

Diese Regeln sollten verinnerlicht und stets befolgt werden! Befinden sich die Daten erst einmal in einem redundanten Zustand oä, ist eine Reparatur nur schwer möglich.

2. Verbindungsaufbau

2.1. PostgreSQL

Mit PostgreSQL Datenbank verbinden:

```
psql -h localhost -U username -W -d datenbankname
      -h: Adresse oder localhost
      -W: Passwort
```

Wichtige Befehle von PostgreSQL

PostgreSQL Command	Erklärung der Commands
\l	Datenbanken anzeigen
\c <Datenbank-Name>	Mit Datenbank verbinden
\i <SQL-Datei>	SQL-File in Datenbank importieren
\q	aus Datenbank aussteigen

Wichtige SQL Befehle

SQL Command	Erklärung der Commands
BEGIN	Einleitung einer neuen Transaktion
SET TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }	Isolations Level einer Transaktion setzten
COMMIT	Beendigung einer erfolgreichen Transaktion
ROLLBACK	Abbruch der laufenden Transaktion

3. Fehlerfälle

„Transaktionen geben eine Isolationsstufe an, mit der definiert wird, bis zu welchem Ausmaß eine Transaktion von Ressourcen- oder Datenänderungen isoliert sein muss, die von anderen Transaktionen durchgeführt werden. Die einzelnen Isolationsstufen werden dahingehend beschrieben, welche Parallelitätsnebeneffekte (wie z. B. Dirty Reads oder Phantomlesezugriffe) zulässig sind.

Das Auswählen einer Isolationsstufe hat keine Auswirkungen auf die Sperren, die zum Schutz der Datenänderung eingerichtet werden!“ [TNE01]

Die Einzelheiten und Unterschiede werden in den weiteren Kapiteln noch genauer erläutert.

Isolations Stufen	Dirty Read	Non-Repeatable Read	Phantom Read
Read uncommitted	Possible	Possible	Possible
Read commited	Not possible	Possible	Possible
Repeatable Read	Not possible	Not possible	Possible
Serializable	Not possible	Not possible	Not Possible

3.1. Dirty Read

3.1.1. Beschreibung

„Dirty Read (anders gesagt „unsauberes Lesen“) tritt auf, wenn eine Transaktion Daten aus einer Tabelle lesen darf, aber jedoch eine weitere Transaktion Daten in der Tabelle ändert, welche noch nicht commited (bestätigt) wurden“ [DOL01] (siehe Beispiel Abb. 2.1).

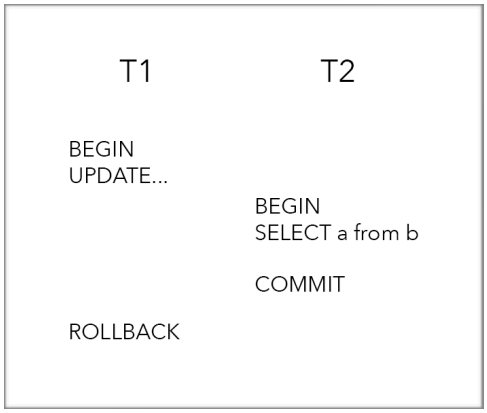


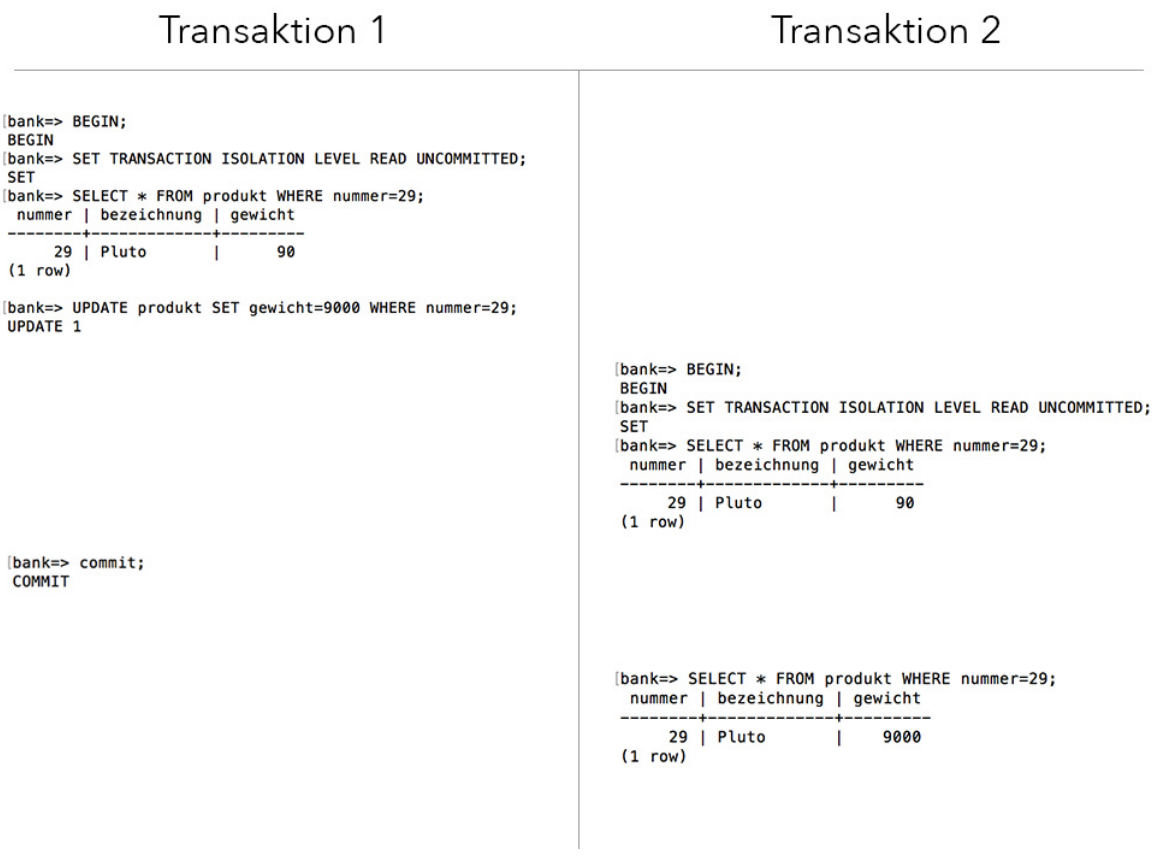
Abb. 2.1: Dirty Read Beispiel

3.1.2. Testung in DBMS

Der „Dirty Read“ wird mit PostgreSQL erzeugt, indem als aller erstes das Isolations Level auf „READ UNCOMMITTED“ gesetzt wird, da sonst ein „Dirty Read“ nicht möglich ist.
Anschließend wird ein Datensatz mit UPDATE auf einen anderen Wert gesetzt.

Die zweite Transaktion wird gestartet und das Isolations Level richtig eingestellt. Mit SELECT wird jetzt der Wert des zuvor, von Transaktion 1 geänderten Datensatzes, ausgelesen. Man stellt fest, dass sich der Wert des Datensatzes nicht geändert hat.

Grund: Transaktion 1 hat noch kein COMMIT abgesendet welches die Werte übergibt.



Schritt-für-Schritt Erklärung

Transaktion 1

Mittels BEGIN wir eine neue Transaktion gestartet

Der Befehl SET TRANSACTION ISOLATION LEVEL ändert das Isolations Level

Zur Kontrolle wird der zu ändernde Datensatz mit SELECT ausgegeben

Mit Hilfe von UPDATE wird der Datensatz geändert

Transaktion 2

Mittels BEGIN wir eine neue Transaktion gestartet

Der Befehl SET TRANSACTION ISOLATION LEVEL ändert das Isolations Level

Zur Kontrolle wird der zu ändernde Datensatz mit SELECT ausgegeben

Transaktion 1

Die Transaktion wird mit COMMIT bestätigt

Transaktion 2

Nach dem COMMIT sieht man mit einem SELECT dass der Datensatz auch in der zweiten Transaktion den „aktuellen“ Wert erhalten hat

3.2. Non repeatable Read

3.2.1. Beschreibung

„Der non repeatabel read tritt beim Auslesen von Daten auf. Die Ausgabe beim Auslesen eines Datensatzes ist nach einem Update (durch eine andere Transaktion) nicht ident, da sie verändert wurde.“ [DOL02] (siehe Abb. 2.1)

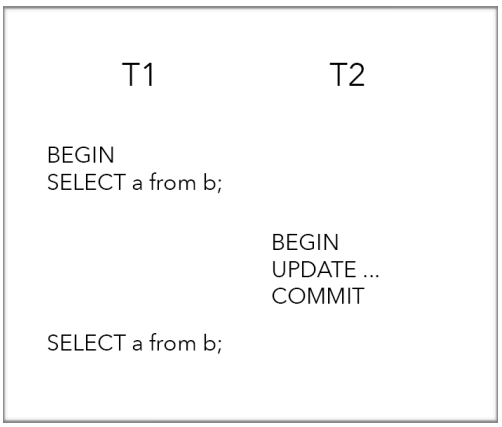


Abb. 2.2: Non repeatable Read Beispiel

3.2.2. Testung in DBMS

Der „Non repeatable Read“ kann erzeugt werden, wenn das Isolations Level auf READ COMMITTED oder READ UNCOMMITTED gesetzt ist, da er sonst nicht möglich ist.
Transaktion 1 startet mit BEGIN eine neue Transaktion für welches anschließend das Isolation Level festgelegt wird. Zum Vergleich wird mittels SELECT ein Datensatz ausgelesen.
Transaktion 2 wird mit BEGIN gestartet und setzt das Isolation Level. Im nächsten Schritt wird mit UPDATE der Datensatz geändert und anschließend mit COMMIT freigegeben.

Wenn jetzt die Transaktion 1 ein SELECT ausführt, wird der geänderte Wert von Transaktion 2 angezeigt. Somit wird gezeigt dass ein SELECT innerhalb einer Transaktion möglicherweise nicht reproduzierbar ist.

Transaktion 1	Transaktion 2

bank=> BEGIN;
BEGIN
bank=> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
bank=> SELECT * FROM produkt WHERE nummer=29;
nummer | bezeichnung | gewicht
--------+-------------+---------
29 | Pluto | 90
(1 row)	

	bank=> BEGIN;
	WARNING: there is already a transaction in progress
	BEGIN
	bank=> COMMIT;
	COMMIT
	bank=> BEGIN;
	BEGIN
	bank=> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
	SET
	bank=> UPDATE produkt SET gewicht=9000 WHERE nummer=29;
	UPDATE 1
	bank=> SELECT * FROM produkt WHERE nummer=29;
	nummer | bezeichnung | gewicht
	--------+-------------+---------
	29 | Pluto | 9000
	(1 row)
	
	bank=> COMMIT;
	COMMIT

bank=> SELECT * FROM produkt WHERE nummer=29; nummer | bezeichnung | gewicht --------+-------------+--------- 29 | Pluto | 9000 (1 row)	

Schritt-für-Schritt Erklärung

Transaktion 1

Mittels BEGIN wir eine neue Transaktion gestartet

Der Befehl SET TRANSACTION ISOLATION LEVEL ändert das Isolations Level

Zur Kontrolle wird der zu ändernde Datensatz mit SELECT ausgegeben

Ein Datensatz wird aktualisiert und mit COMMIT als erfolgreiche Transaktion bestätigt

Transaktion 2

Transaktion 1

Nach einem erneuten SELECT wird ein anderes Ergebnis ausgegeben

3.3. Phantom Read

3.3.1. Beschreibung

Eine Transaktion führt erneut eine Auslegung durch, welche ein Set von Spalten zurück liefert. Mit Hilfe einer Such-Operation wird das erforderliche Suchergebnis zurück geliefert (z.B.: MAX, MIN, SUM…) (siehe Abb. 2.3)

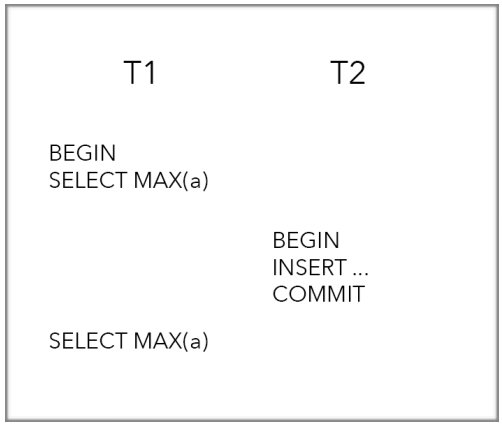


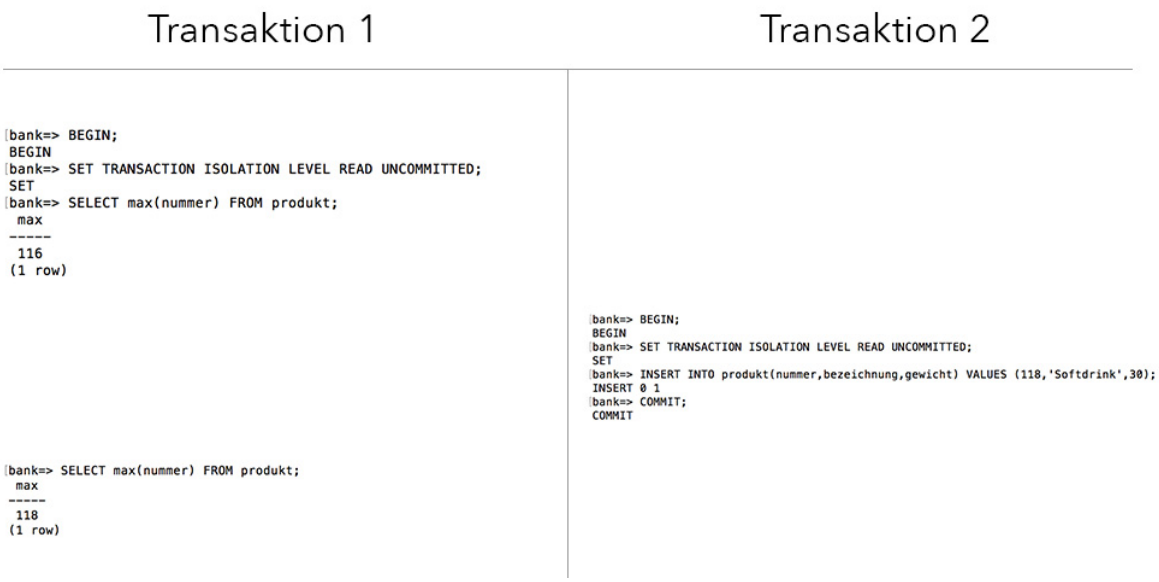
Abb. 2.3: Phantom Read Beispiel

Unterschied zu MySQL:

Phantom Read seit MySQL Version 4.5 nicht mehr möglich!

3.3.2. Testung in DBMS

Durch den Phantom Read tauchen neue Datensätze in einer Tabelle auf, welche von anderen Transaktionen falsch aufgefasst werden.



Schritt-für-Schritt Erklärung

Transaktion 1

Mittels BEGIN wir eine neue Transaktion gestartet

Der Befehl SET TRANSACTION ISOLATION LEVEL ändert das Isolations Level

SELECT max(nummer) … gibt die maximale Nummer aller Datensätze aus

Mittels BEGIN wir eine neue Transaktion gestartet

Der Befehl SET TRANSACTION ISOLATION LEVEL ändert das Isolations Level

INSERT INTO fügt einen neuen Datensatz hinzu

COMMIT bestätigt eine erfolgreiche Transaktion

Transaktion 2

Transaktion 1

SELECT max(nummer) … gibt erneut die aktuelle maximale Nummer aller Datensätze aus

3.4. Lost Update

3.4.1. Beschreibung

Für einen Lost-Update benötigt man zwei oder mehrere Transaktionen.
Transaktion A von z.B. 4 Transaktionen überschreibt den gleichen Datensatz, den Transaktion B geändert hat. Transaktion A überschreibt nun mit seinen neuen Daten für den Datensatz die Daten von Transaktion B und somit gehen die Daten von Transaktion B verloren. Diesen Verlust nennt man Lost-Update.

4. Sperren

Die meisten DBS unterstützen 2 Sperrarten:

- shared Locks (Lesesperren)
  - Ermöglichen paralleles Lesen
  - Verhindern Änderungen
- exclusive Locks (Schreibsperren)
  - Erlauben nur dem Sperrinhaber zu lesen und zu ändern

„Kurz erklärt was der Unterschied zwischen den shared (s) und exclusive (x) locks sind:

Stellen Sie sich vor ein lockable object ist eine Tafel (lockable) in einer Klasse mit einem Lehrer (writer) und einigen Schülern (reader).
Während der Lehrer etwas auf die Tafel schreibt (exclusive lock) …

1.) … kann keiner der Schüler etwas lesen, da der Lehrer etwas aufschreibt und somit die Tafel verdeckt.
—> Wenn ein Objekt exclusive gelockt ist, können shared locks dieses Objekt nicht „lesen, erreichen“.

2.) … können andere Lehrer auch nicht herkommen und etwas auf die Tafel schreiben, weil diese dann unleserlich wird und das verwirrt die Schüler.

—> Wenn ein Objekt exclusive gelockt ist können andere exclusive Locks nicht darauf zugreifen.

Wenn alle Schüler das lesen was auf der Tafel steht (shared lock) …

1.) … können alle Schüler die Informationen gleichzeitig lesen.

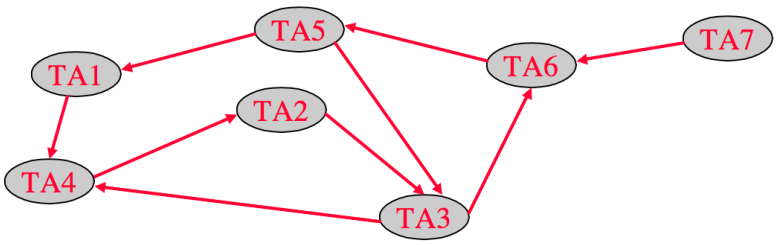
—> Mehrere shared locks können gleichzeitig auf das eine Objekt lesen

2.) … Der Lehrer wartet bis die Schüler fertig gelesen haben, bevor er die Tafel löscht, damit er mehr darauf schreiben kann.

—> Wenn ein oder mehrere shared locks bereits existieren, kann der exclusive lock nicht das Objekt verändern.“ [SHA01]

4.1. Dead-Locks

Bei einem Dead-Lock wartet jede Transaktion auf eine Sperre, welche eine andere Transaktion hält. In eine solche Wartesituationen können beliebig viele Transaktionen involviert sein.



Lösung:

Eine der beteiligten Transaktionen muss zurückgerollt werden.

4.2. Live-Locks

Dieser tritt beispielsweise auf, wenn kleineren Prozessen höhere Prioritäten für die Abarbeitung gesetzt werden und diese dann zu oft auftreten. Die größeren, nicht abgearbeiteten Prozesse werden dabei nicht beachtet, wenn die größeren Prozesse zu länger als 60 Sekunden warten, bekommen sie einen Time-Out und werden anschließend gekillt.

Zitate aus Quellen:

[TNE01] ISOLATIONSTUTEN IM DATENBANKMODUL. MICROSOFT TECHNET [ONLINE] AVAILABLE AT: HTTPS://TECHNET.MICROSOFT.COM/DE-DE/LIBRARY/MS189122(V=SQL.105).ASPX [ABGERUFEN AM 09.12.2015]

[DOL01] DIRTY-READ (2010, MARCH). DATENANKEN ONLINE LEXIKON [ONLINE]. AVAILABLE AT: HTTP://LWIBS01.GM.FH-KOELN.DE/WIKIS/WIKI_DB/INDEX.PHP?N=DATENBANKEN.DIRTY-READ [ABGERUFEN AM 09.12.2015]

[DOL02] NON-REPEATABLE-READ (2010, APRIL). DATENANKEN ONLINE LEXIKON [ONLINE]. AVAILABLE AT: HTTP://LWIBS01.GM.FH-KOELN.DE/WIKIS/WIKI_DB/INDEX.PHP?N=DATENBANKEN.NON-REPEATABLE-READ [ABGERUFEN AM 09.12.2015]

[SHA01] ARJUN SHANKAR. (2012, AUGUST). STACKOVERFLOW [ONLINE]. AVAILABLE AT: HTTP://STACKOVERFLOW.COM/QUESTIONS/11837428/WHATS-THE-DIFFERENCE-BETWEEN-AN-EXCLUSIVE-LOCK-AND-A-SHARED-LOCK [ABGERUFEN AM 09.12.2015]