**Transaktionen**

# **Was sind Transaktionen überhaupt ?**

Das Wort Transaktion kommt aus dem lateinischen ,wobei man achten sollte das trans übersetzt (hin-)über und agere „treiben,handeln,führen“ . Transaktion nennt man Programmschritte ,die als eine logische Einheit betrachtet werden. Die Daten beziehungsweise die Datenbank ist nach diesem Vorgang konsistent.Transaktionen werden entweder vollständig und fehlerfrei oder garnicht ausgeführt.

Bei fehlerhaften Transaktionen werden diese abgebrochen und auf den letzten Zustand zurückgesetzt . => **Rollback**

# **ACID**

ACID beschreibt die Eigenschaften einer Transaktion . Diese muss eingehalten werden, damit eine Transaktion konsistent ist und damit es einzigartig und durchführbar sein muss .

**Atomicity**: Eine Transaktion wird entweder ganz oder gar nicht ausgeführt also sind die Transaktion unteilbar.Falls eine Transaktion abgegbrochen wurde und es zu einem Rollback kommt ist das System unveränderet.

**Consistency:** Nach der Transaktion muss die Datenbank in einer konsistenteM Zustand sein wie am Anfang der Transaktion.

**Isolation:** Mehrere gleichzeitig ausgeführte Transaktionen dürfen sich nicht gegenseite beinflussen

**Durability:**Transaktionen dürfe nicht verloren gehen oder mit der Zeit verblassen. =>dauerhafte Speicherung der Transaktionen auf dem Datenträger.

Diese Regel sollte man befolgen ,falls sich die Daten in einem redudanten Zustand befinden ist es schwer eine Reparatur zu machen .

# **Befehle um eine Transaktionen zu starten**

Begin Beginn einer Transaktion

SET TRANSACTION ISOLATION LEVEL

{ READ UNCOMMITTED

| READ COMMITTED

| REPEATABLE READ

| SERIALIZABLE

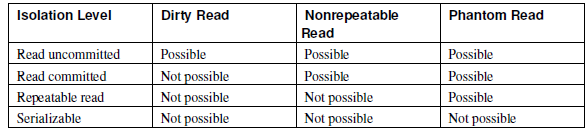
} Setzte einer Isolationstufe einer Transaktion

Commit Beenden einer Transaktion

Rollback laufende Transaktion abbrechen

# **Isolationstufen und ihre Probleme**

Es wird für eine Transaktion eine Isalationsstufe definiert wird welcher Ausmaß von Ressourcen -oder Datenänderungen isoliert sein muss die von einem anderen Transaktionen druchgeführt werden . Diese Isolationsstufen werden beschrieben und welcher diese Fehlerfälle bei den Isolationsstufen auftreten.



# **Non repeatable Read**

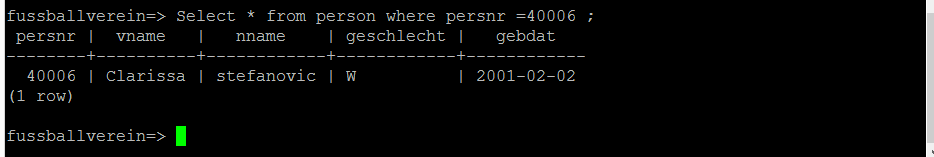
Non repeatable read tritt beim Auslesen einer Datenbank aus. Wie man auf der Tabelle sieht ensteht dieser Problematik wenn man bei der Transaktion die Isalationsstufe auf Readuncommited setzt oder auf Read commited.

Beginn der Transaktion mit dem Befehl **Begin** als nächstes setzt man die Isolationsstufe auf read uncommited oder read commited.

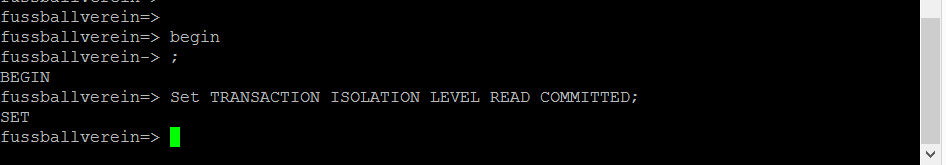
Das macht man auch mit der 2 Transaktion. Außerdem wird dann commited mit dem Befehl **Commit** und wieder mit begin gestartet.Auch die Isolationsstufe von der 2 Transaktion.

In diesem konkretem Beispiel habe ich Read Commited benutzt

## 1 Transaktion

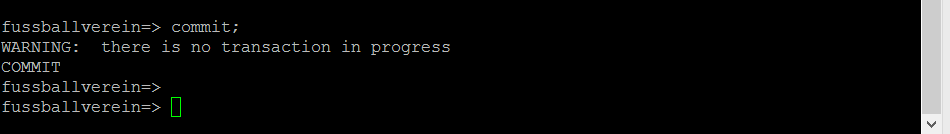
****

Setzen des Isolationslevel;

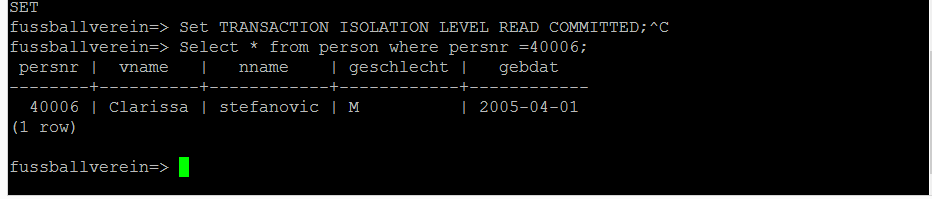
****

## 2 Transaktion

****

****

## Transaktion 1

****

Erklärung: Sobald die 1 Transaktion eine Select abfrage macht bekommt man den wert vom Select der 2 Transaktion.

# **Dirty Read**

Wie man in der Tabelle sieht kommt der Dirty Read nur bei Read uncommitted . Es tritt auf wenn eine Transaktion Daten aus einer Tabelle lesen darf aber eine andere Transaktion die Daten aus der Tabelle ändert die noch nicht commited wurden.

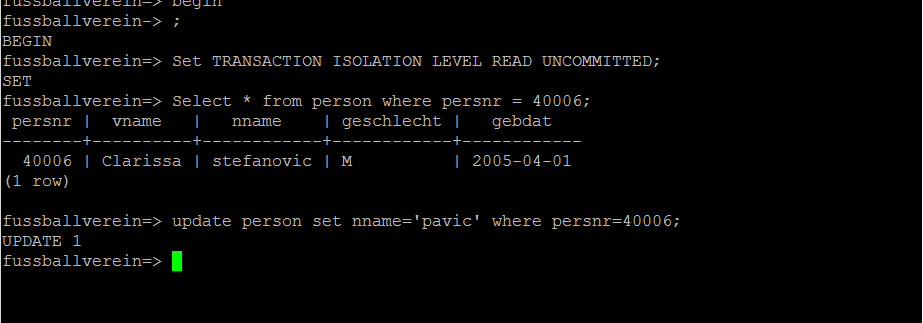
Man setzt eine Transaktion auf Read Uncommitted weil nur dann ist ein Dirty Read möglich .

Wie beim vorigen Beispiel muss man die 1.Transaktion mit dem Befehl begin starten und die Isolationsstufe auf Read uncommitted setzen. Als nächstes muss man eine Änderung eines Datensatzes vornehmen mit Update .

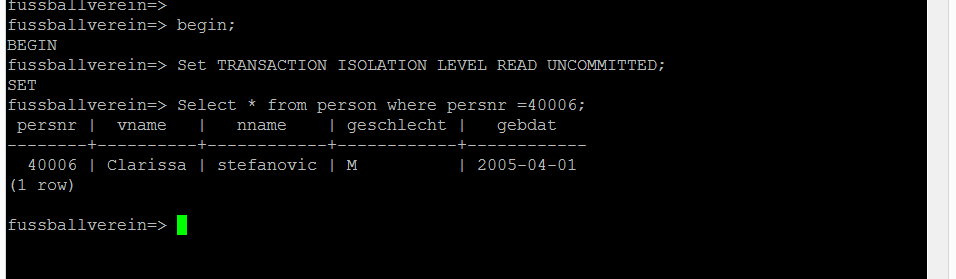
Nebenbei sollte man die 2 Transaktion start und bei ihr auch die Isolationsstufe auf Read uncommited setzen.

Der nächste Schritt ist eine Select abfrage um zu schauen ob die Veränderung übernommen worde .Falls sie nicht übernommen wurde macht man bei dem ersten ein commit und fragt man das gleiche wie oben ab und schaut man ob die Daten verändert worden sind.

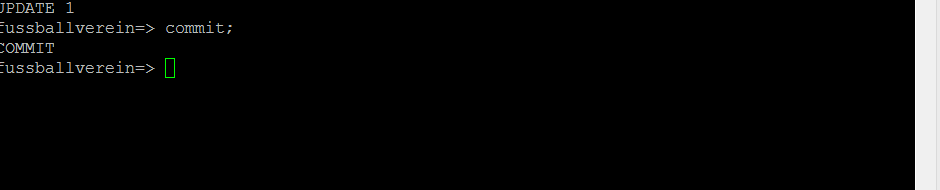
## 1 Transaktion



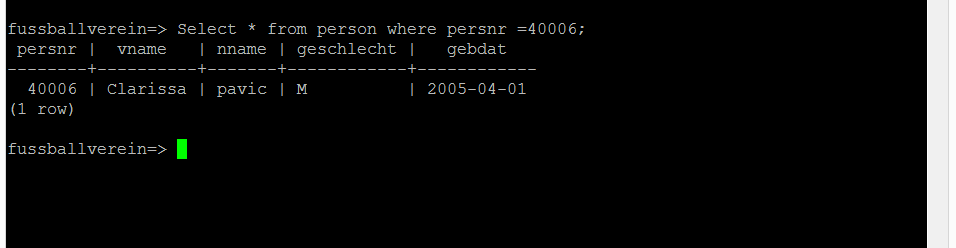
## 2 Transaktion



## 1 Transaktion



## 2 Transaktion



# Phantom Read

Wie man in der Tabelle sieht kommt der Phantom Read in allen Isolationsstufen außer Serializbale vor.

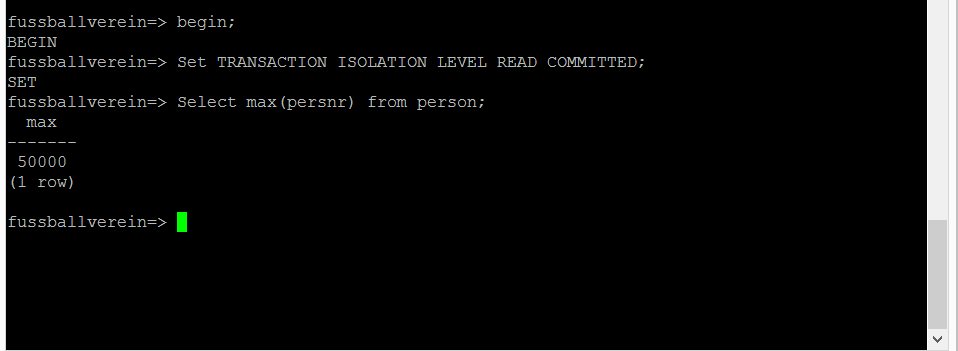
Diese funktioniert erst dann wenn man Aggregationsfunktion bei einem Datensatz benutzt wie zum Beispiel (max,min,sum ..).

Bei diesem konkretem Beispiel startet man die 1 Transaktion und setzt je nach dem wie man will die Isolationsstufe bis auf Serializble. Nach diesen Schritten ließt man den größten INT wert hinaus mit der Aggregation max.

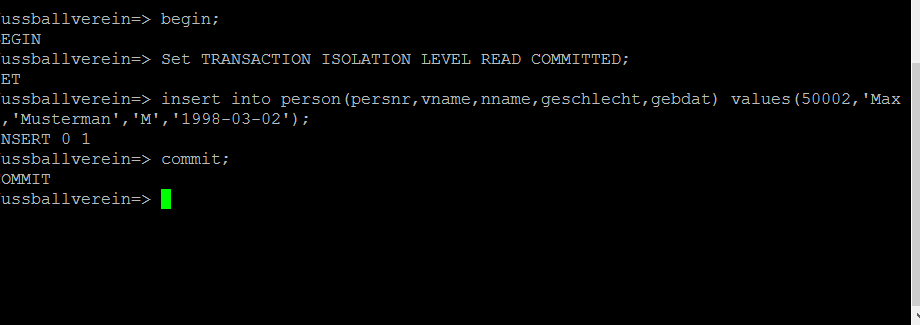
Bei der 2 Transaktion startet man die Transaktion und setzt die Isolationsslevel auf das gewünschte und macht ein Insert wobei der Int wert im neuen Datensatz höher ist als das vorige .Der nächste Schritt ist das man diese Transaktion commited.

Bei der ersten Transaktion führt man den gleichen Select befehl wie vorher aus und schaut ob diese dann größer ist als die vorrige.

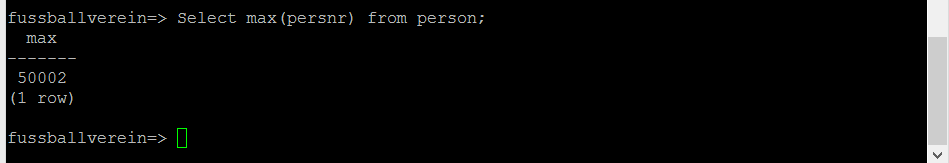
## 1 Transaktion



## 2 Transaktion



## 1 Transaktion



# **Lost Update**

Ein Lost update kann vor kommen wenn mehrere Transaktionen die Daten in der Datenbank veränderen . Wenn zum Beispiel Transaktion A eine Update ausführt aber zur selben Zeit Transaktion B auch ein Update führt also in der Datenbank eine ändereung macht,wird einfach die Änderung von Transaktion A verworfen(überschrieben).

# **Dead Locks**

In einem DeadLock sind mehrer Transaktion involviert welche auf einen Sperre wartet die die andere Transaktion hält.

# **Live Locks**

Bei einem Livelock gibt es mehrer gesperrte Prozesse oder Transaktionen die nicht im selben Status verharren. Sie ändern ständig ihren Zustand. Also wartet nicht ein Prozesses oder bzw. eine Transaktion im wait-zustand.

Anschaulich kann man sich dazu zwei Personen vorstellen, die sich in einem Gang entgegenkommen und fortwährend versuchen, einander in der gleichen Richtung auszuweichen, und sich dabei trotzdem immer gegenseitig blockieren, während bei einem Deadlock sich die zwei Personen nur gegenüberstehen und jeweils darauf warten, dass die andere beiseite geht. =>Sidewalk shuffle