1 Grundproblematik

Aktionen auf einer Datenbank werden erstmal unabhängig voneinander ausgeführt. Dies ist mitunter aber nicht gewünscht. Man möchte mehrere Aktionen zu einer einzigen „Transaktion“ zusammenfassen. Hierbei sind folgende Begriffe, welche das Akronym „**ACID**“ bilden, wichtig:

_____: Zusammenfassung mehrerer DB-Operationen zu einer Transaktion. Nur wenn alle Einzeloperationen erfolgreich abgeschlossen werden, wird die Transaktion als „gültig“ gesetzt – sie wird „commited“. Ansonsten wird sie wieder zurückgestellt – es erfolgt ein „Rollback“.

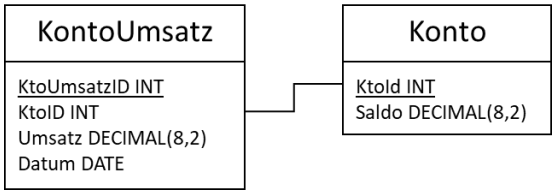
_____: Der Datenbankzustand ist vor und nach der Transaktion konsistent. Die sogenannten „Integritätsbedingungen“ müssen erfüllt sein – welche bspw. die referenzielle Integrität betrifft, aber auch definierte Datenwertverletzungen berücksichtigt (wenn bspw. der Eintrag in „Gewicht“ nicht 0 sein darf)

_____: Datenänderungen einer Transaktion darf nur die Daten eben dieser Transaktion betreffen. Andere, unabhängige Daten, dürfen dadurch nicht beeinflusst oder gesperrt werden.

_____: Daten der Transaktion müssen (auch nach Systemausfällen) stets noch vorhanden sein.

Folgendes Szenario würde bspw. eine Transaktion erfordern:

- Ein User macht auf seinem Konto eine Abbuchung.
- Eine Abbuchung führt zu einem Kontoumsatz.
- Dieser wiederum verändert den Kontostand.



Wir haben also zwei Schreibzugriffe, welche in einer Transaktion zusammengefasst werden müssen. Eine Transaktion beginnt man in MySQL mit `START TRANSACTION` und commited sie mit `COMMIT`. Sollte sie zurückgedreht werden müssen, erfolgt dies mit `ROLLBACK`. Ergänzen Sie nun folgenden Ablauf, so dass die Aktionen atomar durchgeführt werden und die Datenkonsistenz gewahrt wird:

```
SET @myUmsatz = -100;

INSERT INTO KontoUmsatz SET
KtoID = 12345,
Umsatz = @myUmsatz,
Datum = DATE(NOW());

SET @Saldo = (SELECT Saldo FROM Konto WHERE KtoID = 12345);

UPDATE Konto SET
Saldo = @Saldo + @myUmsatz
WHERE KtoID = 12345;
```

Aus der SQL-Sequenz ergeben sich nun aber potentielle Probleme. Gehen wir von folgendem Szenario aus:

- DB-Client 1 erzeugt auf dem Konto einen Umsatzeintrag
- DB-Client 2 erzeugt auf dem Konto einen weiteren Umsatzeintrag
- Beide Clients selektieren den Saldo für das Update und erhalten den gleichen Wert
- Beide Clients erhöhen nun den Saldo um den Betrag und schreiben den Wert zurück

Nun würde aber nur der Wert des letzten Clients, der zurückgeschrieben hat, übernommen werden. Dies bedeutet aber eine Verletzung der Isolation und sollte somit von der Datenbank verhindert werden. Ob dies nun tatsächlich von der Datenbank im Rahmen einer Transaktion verhindert wird, wollen wir nun überprüfen. Hierfür gehen wir nun wie folgt vor:

- Erzeugung der beiden Tabellen.
- Initialer Datensatz für das Konto ist `KtoID = 12345, Saldo = 1000.0;`
- `START TRANSACTION` und Einfügen des Kontoumsatzes wie oben beschrieben.
- `UPDATE` des Kontostandes auf dem 1. Clients.
- Öffnen eines neuen MySQL Clients und `SELECT *` auf Kontenumsatz
- `SELECT *` auf Kontenumsatz auf dem 1. und auf dem 2. Client.
- Versuch im 2. Client, einen weiteren Kontenumsatz inklusive Kontenupdate zu schreiben.
- `COMMIT` der Transaktion auf dem 1. Client.
- `SELECT *` auf Konto auf dem 1. und auf dem 2. Client.

Was beobachten Sie beim `SELECT` auf dem Kontosaldo der beiden Clients:

`SELECT` auf Kontensaldo:	**Client 1**	**Client 2**
Vor dem `COMMIT`		
Nach dem `COMMIT`		

Wie verhält sich der `INSERT` auf Client 2 vor und nach dem `COMMIT`?

Das bedeutet, dass ein in einer Transaktion manipulierter Datensatz während dieser Transaktion für Änderungen durch andere Clients **gesperrt** ist!

Nun prüfen wir das Verhalten bei einem `ROLLBACK`:

- `START TRANSACTION` und Einfügen des Kontoumsatzes wie oben beschrieben.
- `UPDATE` des Kontostandes auf dem 1. Clients.
- Im 2 Client `SELECT *` auf Kontenumsatz
- `SELECT *` auf Kontenumsatz auf dem 1. und auf dem 2. Client.
- `ROLLBACK` der Transaktion auf dem 1. Client.
- `SELECT *` auf Kontenumsatz und Konto auf dem 1. und auf dem 2. Client.

SELECT auf Kontensaldo:	**Client 1**	**Client 2**
Vor dem `ROLLBACK`		
Nach dem `ROLLBACK`		