Was sind Transaktionen?

- Verbund von Anweisungen die atomar ausgeführt werden.
- Die Datenbank bleibt konsistent bei auftretenden Verarbeitungsfehlern.
- Vergleichbar mit Semaphoren und Monitoren in der Betriebssystemtechnik.
  - Allerdings sind parallele SQL Anweisungen möglich, dank des Historie-Mechamismus und von Transasktionsmonitoren.
- Müssen nach dem **ACID** Modell implementiert sein!

Was sind Transaktionen?

- **ACID**
  - **Atomicity**: Transaktionen sind „unteilbar“. Bei Abbruch, ist das System unverändert.
  - **Consistency**: Nach Ausführung der Transaktion muss der Datenbestand in einer konsistenten Form sein, wenn er es bereits zu Beginn der Transaktion war.
  - **Isolation**: Transaktionen dürfen sich diese nicht gegenseitig beeinflussen.
  - **Durability**: Die Auswirkungen einer Transaktion müssen im Datenbestand dauerhaft bestehen bleiben.

Was sind Transaktionen?

- Es kann nicht alles in Transaktionen ausgeführt werden! Dies sind DDL-Anweisungen (Data Definition Language).
  Anweisungen:
  - zum Datenbank erstellen/löschen
  - Tabellen erstellen/löschen/ändern
  - gespeicherte Routinen erstellen/löschen/ändern

Was sind Transaktionen?

- Transaktionen werden im Grunde durch 3 Anweisungen beschrieben:
  - **BEGIN**, beginnt eine Transaktion.
  - **COMMIT**, Änderungen werden übernommmen.
  - **ROLLBACK**, alle Modifikationen werden rückgängig gemacht.
- Mit SELECT kann schon ein Stand gelesen werden der noch nicht mit COMMIT bestätigt wurde.

Was sind Transaktionen?

SQL Anweisungen:

```
USE bank;
BEGIN;
INSERT INTO kunde ( name, vorname ) VALUES ( 'Gruenwoldt', 'Lutz' );
SELECT * FROM kunde;
ROLLBACK;
SELECT * FROM kunde;
```

```
USE bank;
BEGIN;
INSERT INTO kunde ( name, vorname ) VALUES ( 'Ries', 'Christian' );
SELECT * FROM kunde;
COMMIT;
SELECT * FROM kunde;
```

Transaktionen in MySQL

- MySQL unterstützt Transaktionen mit den MEMORY-Speicher-Engines: InnoDB & BDB.

```
CREATE TABLE `bank`.`kunde` (
`id` INT NOT NULL AUTO_INCREMENT ,
`name` VARCHAR( 40 ) NOT NULL ,
`vorname` VARCHAR( 40 ) NOT NULL ,
PRIMARY KEY ( `id` )
) ENGINE = InnoDB
```

Transaktionen in MySQL

Beispiel 2

Siehe dazu die 2 Dateien ”auto_increment_example_1.php” & ”auto_increment_example_2.php”. Diese Dateien beinhalten denselben Quellcode. Die 1. Datein enthällt zudem noch Kommentare zum Programm.

Das Beispiel benötigt zwei Tabellen:

```
CREATE TABLE `konto` (
  `id` int(11) NOT NULL auto_increment,
  `uid` int(11) NOT NULL,
  `kontonr` int(11) NOT NULL,
  `guthaben` decimal(10,0) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB;
```

```
CREATE TABLE `kunde` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(40) NOT NULL,
  `vorname` varchar(40) NOT NULL,
  `TESTID` int(11) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB;
```

Transaktionen in MySQL

- Verwendung:
  Es wird Programm 1 aufgerufen, dieses geht in Schlafmodus für 10 Sekunden. In dieser Zeit sollte Programm 2 aufgerufen werden. Beide Programme erstellen einen neuen Kunden in der Tabelle kunde. Nach dem Warten von 10 Sekunden wird die letzte eingetragene Kunden ID ermittelt und mit dieser die Verknüpfung zum Konto erstellt.

Transasktionen in MySQL

- Ergebnis:
  Wenn keine Transaktionen verwendet werden, wird dem Konto die falsche ID des Kunden zugewiesen. Da in den 10 Sekunden Wartezeit ein anderer Prozess einen neuen Kunden einträgt und diese dann für beide Konten verwendet wird.

  Wenn Transaktionen verwendet werden, dann werden zwei von einander unabhängige Konten erstellt. So muss es sein.

Weitere MySQL Möglichkeiten

- SAVEPOINT und ROLLBACK TO SAVEPOINT
  - Es kann zu bestimmten Zuständen gesprungen werden die mit SAVEPOINT markiert wurden.
- LOCK TABLES und UNLOCK TABLES
  - Tabellen können für andere Threads/Prozesse gesperrt werden.