



# DB: Datenbanken

## Transaktionen

Prof. Dr. Ludger Martin

# Gliederung

---

- ★ Einführung
- ★ Transaktionen
- ★ Savepoints

# Einführung

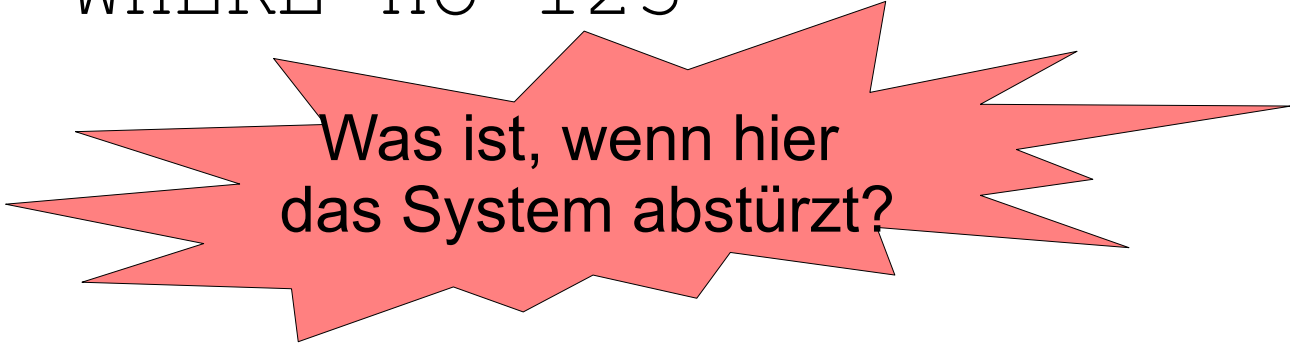
---

- ★ Kapselung mehrerer SQL-Kommandos als Transaktion
- ★ Transaktionen stellen sicher, dass Gruppen von SQL-Kommandos vollständig oder gar nicht ausgeführt werden
- ★ Sicherstellung, dass Daten nicht gleichzeitig von anderen Benutzern verändert werden (InnoDB zeilenweise)  
→ Datensystem sicherer machen

# Einführung

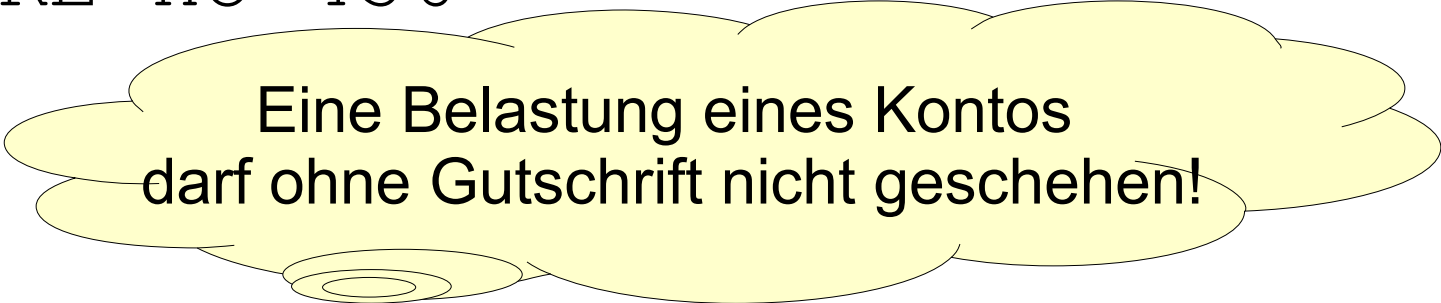
## ★ Beispiel Überweisung:

```
UPDATE account SET value=value-100  
WHERE no=123
```



Was ist, wenn hier  
das System abstürzt?

```
UPDATE account SET value=value+100  
WHERE no=456
```



Eine Belastung eines Kontos  
darf ohne Gutschrift nicht geschehen!

# Einführung

---

## ★ ACID:

Atomicity, Consistency, Isolation, Durability  
(von InnoDB eingehalten)

★ **Atomicity:** Unteilbarkeit von Transaktionen  
(Transaktionen vollständig oder gar nicht ausführen)

★ **Consistency:** Am Ende einer Transaktion ist die DB in konsistentem (fehlerfreien) Zustand.  
Wenn eine Transaktion zur Verletzung von Gültigkeitsregeln führt, muss sie abgebrochen werden.

# Einführung

---

## ★ ACID: (Fortsetzung)

★ **Isolation:** Mehrere gleichzeitige Transaktionen dürfen sich nicht beeinflussen bzw. stören

★ Transaktion sieht DB immer im Anfangszustand (außer Änderungen durch Transaktion)

★ Isolierung geht auf Kosten der Geschwindigkeit

★ **Durability:** Wenn Transaktion fertig, müssen Daten gespeichert sein!

★ Auch wenn es unmittelbar danach zum Absturz kommt

★ Schreiben von Protokolldateien, auch auf Kosten der Performance

# Transaktionen

---

- ★ MySQL/MariaDB (nur InnoDB) arbeitet generell im Auto-Commit-Modus  
`SET AUTOCOMMIT=1` bzw. `0`
- ★ Starten einer Transaktion mit `BEGIN` oder `START TRANSACTION`
- ★ Beende mit `COMMIT` (bestätigen) oder `ROLLBACK` (widerrufen)

# Transaktionen

---

- ★ Keine hierarchischen Transaktionen möglich
- ★ Wird Verbindung zu Datenbank vor Transaktionsende beendet, wird Transaktion widerrufen
- ★ Transaktionen automatisch durch `ALTER TABLE`, `CREATE TABLE`, `DROP`, `LOCK/UNLOCK`, `TRUNCATE` **akzeptiert**



# Transaktionen

## Beispiel

---

### ★ Tabelle:

```
CREATE TABLE account (  
    no INT NOT NULL,  
    value INT NOT NULL,  
    PRIMARY KEY (no)  
) ENGINE = InnoDB;
```

# Transaktionen

---

## ★ Beispiel Überweisung:

```
START TRANSACTION;
```

```
UPDATE account SET value=value-100  
WHERE no=123;
```

```
UPDATE account SET value=value+100  
WHERE no=456;
```

```
COMMIT;
```

# Transaktionen

## Beispiel

### Verbindung A

★ INSERT INTO account  
VALUES (1, 10);

★ INSERT INTO account  
VALUES (2, 20);

★ SELECT \* FROM account;

no	value
1	10
2	20

★ START TRANSACTION;

### Verbindung B

# Transaktionen

## Beispiel

### Verbindung A

★ UPDATE account SET  
value=11 WHERE no=1;

★ SELECT \* FROM account;

no		value
1		11
2		20

### Verbindung B

★ SELECT \* FROM account;

no		value
1		10
2		20

★ START TRANSACTION;

# Transaktionen

## Beispiel

### Verbindung A

★ SELECT \* FROM account;

no		value
1		11
2		20

★ COMMIT;

### Verbindung B

★ UPDATE account SET  
value=21 WHERE no=2;

★ SELECT \* FROM account;

no		value
1		10
2		21

★ UPDATE account  
SET value=value+3  
WHERE no=1;

# Transaktionen

## Beispiel

### Verbindung A

★ SELECT \* FROM account;

no		value
1		11
2		20

★ SELECT \* FROM account;

no		value
1		11
2		20

### Verbindung B

★ SELECT \* FROM account;

no		value
1		14
2		21

★ ROLLBACK;

★ SELECT \* FROM account;

no		value
1		11
2		20

# Savepoints

---

- ★ Innerhalb einer Transaktion können benannte Savepoints definiert werden: `SAVEPOINT name`
- ★ Sobald `ROLLBACK TO SAVEPOINT name` aufgerufen wird, werden alle Kommandos bis zum Savepoint akzeptiert und alle danach widerrufen.
- ★ Savepoints können nur innerhalb einer Transaktion genutzt werden
- ★ Durch `COMMIT` oder `ROLLBACK` werden alle Savepoints gelöscht

# Savepoints

## Beispiel

★ START TRANSACTION;

★ SELECT \* FROM account;

+	-----	+	-----	+
	no		value	
+	-----	+	-----	+
	1		11	
	2		20	
+	-----	+	-----	+

★ UPDATE account SET value=12 WHERE no=1;

★ SAVEPOINT one;

★ UPDATE account SET value=21 WHERE no=2;



# Savepoints Beispiel

★ `SELECT * FROM account;`

no	value
1	12
2	21

★ `ROLLBACK TO SAVEPOINT one;`

★ `SELECT * FROM account;`

no	value
1	12
2	20

★ `COMMIT;`

# Literatur

---

- ★ Kofler, Michael: MySQL 5, 3. Auflage, Addison-Wesley, 2005
- ★ Vossen, Gottfried: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, 5. Auflage, Oldenburg Wissenschaftsverlag, 2008
- ★ Lubkowitz, M: Webseiten programmieren und gestalten, Galileo Press, 2004
- ★ Oracle: MySQL 5.7 Reference Manual,  
<https://dev.mysql.com/doc/refman/5.7/en/>