

# Datenbanksysteme

SQL DAL

Jan Haase

2024

Abschnitt 10

---

# Wiederholung: Kategorien von SQL-Befehlen

- **DQL (Data Query Language)**  
*Abfrage und Zusammenstellung von Daten*

- SELECT

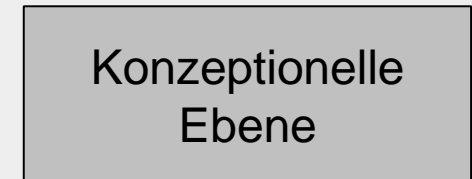
- **DML (Data Manipulation Language)**  
*Umgang mit Tabelleninhalten*

- INSERT
- UPDATE
- DELETE

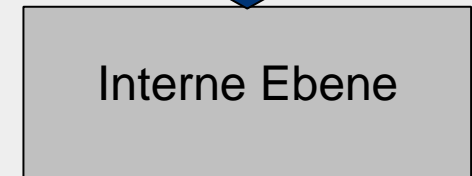


- **DDL (Data Definition Language)**  
*Erstellen und Ändern von Datenbanken und Tabellen*


- CREATE
- ALTER
- DROP



- **DAL (Data Administration Language)**
  - TCL (Transaction Control Language)
  - DCL (Data Control Language)



# Themenübersicht

- Grundbegriffe und Datenbankentwurf
  - Entity-Relationship-Modelle
  - Relationales Datenbankmodell
  - Normalisierung
  - Arbeiten mit relationalen Datenbanken (SQL)
    - DQL
    - DML
    - DDL
-  **DAL (Data Administration Language)**

# Themenübersicht

- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle
- Relationales Datenbankmodell
- Normalisierung
- Arbeiten mit relationalen Datenbanken (SQL)
  - DQL
  - DML
  - DDL
  - DAL (Data Administration Language)
    - ➔ **TCL (Transaction Control Language)**

# Transaktionen – Motivation und Definition

## Motivation

- In großen Datenbanksystemen kommt es zu (fast) gleichzeitigen Zugriffen verschiedener Nutzer auf dieselben Datenbanktabellen.
- Das kann zu Problemen mit der Datenkonsistenz führen.
- Probleme entstehen auch bei Programmabstürzen oder Stromausfällen.
- Das DBMS sollte dafür sorgen, dass kritische Zustände vermieden werden und kein Nutzer auf andere Nutzer Rücksicht nehmen muss.
- Dazu dienen u. a. Transaktionen.

## Definition: Transaktion

- Eine Transaktion ist eine feste Folge von Datenbankoperationen, die eine logische Einheit bilden.
- Diese sollen komplett ausgeführt werden, ohne dass nebenläufige Operationen von anderen Nutzern stören.
- Transaktionen beziehen sich auf die Veränderung von Tabelleninhalten, nicht auf das Erzeugen oder Löschen von Tabellen.

## Transaktionen – ACID-Merkmale

Mit dem Begriff der Transaktion sind vier Merkmale verbunden  
(**ACID-Eigenschaften**):

- **Atomicity** (Atomarität, Unteilbarkeit)  
Eine Aktualisierung wird entweder komplett oder gar nicht ausgeführt.
- **Consistency** (Konsistenz)  
Jede Aktualisierung führt von einem konsistenten (d. h. logisch fehlerfreien Datenbankzustand) in einen konsistenten Zustand über.
- **Isolation**  
Jede Aktualisierung wird so ausgeführt, als würde sie die Datenbank alleine nutzen.
- **Durability** (Dauerhaftigkeit)  
Der von einer abgeschlossen Aktualisierungsoperation bewirkte Datenbankzustand ist dauerhaft.

# Anomalien

## ● DIRTY READ

- Transaktion T1 transferiert 300 € von Konto A nach Konto B, wobei zunächst Konto A belastet wird und danach die Gutschrift auf Konto B erfolgt. T1 wird aber vorzeitig, vor dem Schreiben auf Konto B abgebrochen.
- Transaktion T2 schreibt gleichzeitig dem Konto A 3% Zinseinkünfte gut.

| Schritt | T1   | T2  |
|---------|--|---|
| 1.      | <code>read (A, a<sub>1</sub>)</code>             |   |
| 2.      | <code>a<sub>1</sub> = a<sub>1</sub> - 300</code> |   |
| 3.      | <code>write (A, a<sub>1</sub>)</code>            |   |
| 4.      |  | <code>read (A, a<sub>2</sub>)</code>              |
| 5.      |  | <code>a<sub>2</sub> = a<sub>2</sub> * 1.03</code> |
| 6.      |  | <code>write (A, a<sub>2</sub>)</code>             |
| 7.      | <code>read (B, b<sub>1</sub>)</code>             |   |
| 8.      | ...  |   |
| 9.      | <code>abort</code>                               |   |

# Anomalien

## ● LOST UPDATE

- Transaktion T1 transferiert 300 € von Konto A nach Konto B, wobei zunächst Konto A belastet wird und danach die Gutschrift auf Konto B erfolgt.
- Transaktion T2 schreibt gleichzeitig dem Konto A 3% Zinseinkünfte gut.

| Schritt | T1   | T2  |
|---------|--|---|
| 1.      | <code>read (A, a<sub>1</sub>)</code>             |   |
| 2.      | <code>a<sub>1</sub> = a<sub>1</sub> - 300</code> |   |
| 3.      |  | <code>read (A, a<sub>2</sub>)</code>              |
| 4.      |  | <code>a<sub>2</sub> = a<sub>2</sub> * 1.03</code> |
| 5.      |  | <code>write (A, a<sub>2</sub>)</code>             |
| 6.      | <code>write (A, a<sub>1</sub>)</code>            |   |
| 7.      | <code>read (B, b<sub>1</sub>)</code>             |   |
| 8.      | <code>b<sub>1</sub> = b<sub>1</sub> + 300</code> |   |
| 9.      | <code>write (B, b<sub>1</sub>)</code>            |   |



# Anomalien

## ● PHANTOMPROBLEM

- Transaktion T2 berechnet zweimal die Summe der Kontostände aller Konten.
- Zwischen der ersten und zweiten Abfrage schreibt Transaktion T1 einen neuen Datensatz in die Tabelle Konten.
- In der zweiten Abfrage von T2 tauchen nun Ergebnisse auf, die in der ersten nicht sichtbar waren.

| Schritt | T1  | T2  |
|---------|---|---|
| 1.      |   | <b>SELECT SUM(KontoStand)<br/>FROM Konten</b> |
| 2.      | <b>INSERT INTO Konten<br/>VALUES (C,1000,...)</b> |   |
| 3.      |   | <b>SELECT SUM(KontoStand)<br/>FROM Konten</b> |

# Übersicht: Befehle zur Transaktionssteuerung

- **Transaktionsmanager** (Komponente des DBMS) kann mit SQL-Anweisungen gesteuert werden:
  - **BEGIN TRANSACTION**

Start einer Transaktion.

*Achtung, bei ORACLE beginnt der Transaktionsbeginn implizit!*
  - **COMMIT**

Die vorher eingegebenen Befehle werden endgültig in die Datenbank übernommen.
  - **ROLLBACK**

Die Transaktion wird explizit (per Programm) oder implizit (Fehlersituation bei Commit) abgebrochen. Änderungen, die seit dem letzten Ende einer Transaktion stattgefunden haben, werden verworfen.
  - **SAVEPOINT**

Ein Sicherungspunkt wird gesetzt. Mit dem nächsten ROLLBACK werden nur die SQL-Befehle verworfen, die nach dem Sicherungspunkt ausgeführt wurden.

Transaktionen dürfen nicht verschachtelt werden.

Es lassen sich aber mehrere Sicherungspunkte in ORACLE erstellen.

## Transaktionen – Beispiel (1/2)

- In der Tabelle Konten werden Geldbeträge gespeichert:  
`CREATE TABLE Konten`  
    `(Name VARCHAR2(20), Kontostand NUMBER);`  
`INSERT INTO Konten VALUES ('Meyer', 1000);`  
`INSERT INTO Konten VALUES ('Schulze', 1000);`
- Die Summe der beiden Konten muss immer 2000 betragen.

- Erfolgreiche Überweisung:

```
UPDATE Konten SET Kontostand=Kontostand-100
WHERE Name='Meyer';

UPDATE Konten SET Kontostand=Kontostand+100
WHERE Name='Schulze';

COMMIT;
```

## Transaktionen – Beispiel (2/2)

- Abgebrochene Überweisung:

```
UPDATE Konten SET Kontostand=Kontostand-100  
WHERE Name='Meyer' ;
```

```
UPDATE Konten SET Kontostand=Kontostand+100  
WHERE Name='Schulze' ;
```

```
ROLLBACK ;
```

- Abbruch der Überweisung nach SAVEPOINT:

```
UPDATE Konten SET Kontostand=Kontostand-100  
WHERE Name='Meyer' ;
```

```
SAVEPOINT S1 ;
```

```
UPDATE Konten SET Kontostand=Kontostand+100  
WHERE Name='Schulze' ;
```

```
ROLLBACK TO SAVEPOINT S1 ;
```

# Übungsaufgabe Transaktionen

- Ein Nutzer A führt auf einer Datenbank folgende SQL-Anweisungen aus:

```
INSERT INTO T1 (Name, Alt) VALUES ('Heinz', 42);  
UPDATE T1 SET Alt = Alt+1;  
COMMIT;
```

- Ein zweiter Nutzer B führt auf der gleichen Datenbank folgende SQL-Befehle aus:

```
INSERT INTO T1 (Name, Alt) VALUES ('Verena', 33);  
UPDATE T1 SET Alt = Alt+1;  
COMMIT;
```

Gehen Sie davon aus, dass die Tabelle T1 mit den Spalten Name und Alt erfolgreich angelegt wurde und leer ist, bevor A und B tätig werden.

- Aufgaben:
  - a) Welche Endzustände können in der Tabelle T1 erreicht werden, wenn die Datenbank keine Transaktionssteuerung hat? Dabei seien einzelne INSERT und UPDATE nicht unterbrechbar.
  - b) Welche Endzustände können in der Tabelle T1 erreicht werden, wenn die Datenbank eine vollständige Transaktionssteuerung hat?

# Zusammenfassung

- TCL (Transaction Control Language)
  - Motivation / Definition
    - Atomicity
    - Consistency
    - Isolation
    - Durability
  - Anomalien
    - Lost Update
    - Dirty Read
    - Phantomproblem
  - Transaktionsmanager
    - BEGIN TRANSACTION
    - COMMIT
    - ROLLBACK
    - SAVEPOINT