

# MySQL

## Einführung

Zellescher Weg 12

Willers-Bau A116

Tel. +49 351 - 463 - 39871

Guido Juckeland ([guido.juckeland@tu-dresden.de](mailto:guido.juckeland@tu-dresden.de))

# Inhalt

---

- Grundsätzliches zu Datenbanken
- Modellierung von Daten
- Datenbank anlegen
- Tabellen anlegen
- Daten einfügen
- Abfragen auf Tabelleninhalte
- Abfragen über mehrere Tabellen
- Tabellen leeren und löschen

# Datenbanken (1)

---

- geordnete Datensammlung
- relationale Datenbank basiert auf Relationenalgebra
  - Selektion
  - Projektion
  - Kreuzprodukt
  - Vereinigung / Differenz
- SQL (Structured Query Language)
- Schritte zur Datenbank
  - Datenanalyse, Modellierung
  - Entwurf des Datenbank-Schemas
  - Erstellen und Füllen der Datenbank
  - Abfragen

# Datenbanken (2)

---

- relationale Datenbank
  - bestehend aus Tabellen
  - zweidimensionale Strukturen
  - eines oder eine Kombination von Attributen bilden den Schlüssel
  - in den Zeilen sind Datensätze
  - Spaltenüberschriften beschreiben die Felder der einzelnen Datensätze
  - Primär- und Fremdschlüssel verbinden die Daten logisch miteinander
  - Beziehungen zwischen Daten können auch durch Tabellen dargestellt werden
- 
- Eine gute Modellierung ist der Schlüssel zum Erfolg bei der Arbeit mit Datenbanken

# Datenanalyse

---

- Was soll gespeichert werden?
  - Beispiel TU-Mitarbeiter: Name, Vorname, Institut mit Anschrift, Mailadressen, Kostenstelle, Geburtsdatum, ausgeliehene Bücher
- Welche Beziehungen haben diese Daten untereinander?
  - eins-zu-eins
  - eins-zu-viele
  - viele-zu-viele
- daraus kann man oft schon ein Datenbankmodell ableiten
- vermieden werden sollte:
  - doppelte Beziehungen
  - mehrfache Speicherung der selben Daten
  - Anomalien
- Theorie: Normalisierung

# Tabellen – einfaches Beispiel (eins-zu-eins)

The diagram shows a table with four columns: 'Nummer', 'Vorname', 'Name', and 'Institut'. The 'Nummer' column is highlighted in dark yellow and labeled 'Schlüssel' (Key). The 'Vorname', 'Name', and 'Institut' columns are highlighted in light yellow and labeled 'Attribute'. The third row is highlighted in light purple and labeled 'Datensatz' (Data Record). The table contains the following data:

Nummer	Vorname	Name	Institut
1	Karl	Meier	Chemie
2	Hans	Müller	Mathematik
3	Werner	Böhm	Mathematik
...	...	...	...

# Tabellen – Fremdschlüssel (eins-zu-viele)

Nummer	Vorname	Name	Institutsnummer
1	Karl	Meier	1
2	Hans	Müller	2
3	Werner	Böhm	2
...	...	...	...

Fremdschlüssel

Nummer	Name	Ort	PLZ
1	Chemie	Dresden	01146
2	Mathematik	Dresden	03423
3	Mathematik	München	99110
...	...	...	...

# Tabellen – viele-zu-viele

Nummer	Vorname	Name	Institutsnummer
1	Karl	Meier	1
2	Hans	Müller	2
3	Werner	Böhm	2

Mitarbeiter	Buch
1	1
2	3
2	1

Kombination aller Attribute bildet üblicherweise den Schlüssel

ID	Titel	Autor	Exemplare
1	SQL	www.teia.de	3
2	SQL Kochbuch	Paul DuBois	1
3	Linux-Kernel-Handbuch	Robert Love	1

# Normalformen

---

- erste Normalform:
  - keine Duplikate in den Zeilen
  - pro Spalte nur gleiche Werte
  - jede Zelle nur einen Wert (nicht Titel + Autor zusammen)
- zweite Normalform:
  - alle Attribute, die nicht Schlüssel sind, hängen vom gesamten Primärschlüssel ab
- dritte Normalform:
  - keine transitiven Abhängigkeiten, d.h. ein Eintrag in einer Spalte, die keine Schlüssel ist, darf nicht von einem Eintrag einer anderen Spalte, die keine Schlüssel ist, abhängen

# MySQL Syntax

---

- Schlüsselwörter CREATE, SELECT, INSERT, DELETE, ...
- Spaltennamen dürfen keine Schlüsselwörter sein
- Zeichenketten, die nicht in Hochkommas eingefasst sind, werden als Spaltennamen aufgefasst
- bei Verwendung des Kommandozeileninterfaces müssen Kommandos mit einem Semikolon abgeschlossen werden
- wenige Schritte zur Nutzung
  - mysql installieren
  - für Nutzer root ein Passwort setzen
  - Datenbank & Tabellen anlegen
  - Nutzer mit Passwort für diese Datenbank anlegen

# Tabelle anlegen und Daten einfügen - einfaches Beispiel

---

- Nach Installation muss für den root-Nutzer ein Passwort vergeben werden:

```
mysqladmin -u root password <neuespassword>
```

- Kommandozeilenzugriff auf die Datenbank:

```
mkluge> mysql -u <nutzer> -p <datenbank>
```

- CREATE DATABASE name ;

- USE name ;

- CREATE TABLE ... ;

- INSERT INTO tabelle (feld1,feld2) VALUES (wert1,wert2) ;

# Rechte auf Datenbanken verteilen

---

- Kommandozeile:

```
mysql -u root -p <datenbank>
```

- GRANT ALL ON <datenbank>.\* TO '<nutzer>'@'<rechner>' IDENTIFIED BY '<passwort>'

- statt ALL: SELECT, INSERT, UPDATE, ...

# Tabellen anlegen (1)

---

- zuerst Datenbank mit `CREATE DATABASE <dbname>;`
- `CREATE TABLE mitarbeiter (  
    id          INTEGER NOT NULL PRIMARY KEY,  
    name       VARCHAR(200) NOT NULL,  
    mail       VARCHAR(200),  
    ... weitere Definitionen ...  
);`
- mögliche weitere Definitionen:
  - `UNIQUE(feldnamen)`
  - `FOREIGN KEY(feldname) REFERENCES anderetab(feld)`
  - `PRIMARY KEY(feld1,feld2)`
- Tabelle müssen hierarchisch "von unten" her angelegt werden, d.h. Tabellen ohne Fremdschlüssel zuerst

# Tabellen anlegen (2)

Nummer	Vorname	Name	Institutsnummer
1	Karl	Meier	1
2	Hans	Müller	2
3	Werner	Böhm	2
...	...	...	...

Fremdschlüssel

Nummer	Name	Ort	PLZ
1	Chemie	Dresden	01146
2	Mathematik	Dresden	03423
3	Mathematik	München	99110
...	...	...	...

# Tabellen anlegen (3)

Nummer	Name	Ort	PLZ
1	Chemie	Dresden	01146
2	Mathematik	Dresden	03423
3	Mathematik	München	99110
...	...	...	...

```
CREATE TABLE institut (  
    nummer    INTEGER NOT NULL PRIMARY KEY,  
    name      VARCHAR(50),  
    ort       VARCHAR(50),  
    plz       INTEGER  
);
```

# Tabellen anlegen (4)

Nummer	Vorname	Name	Institutsnummer
1	Karl	Meier	1
2	Hans	Müller	2
3	Werner	Böhm	2
...	...	...	...

Fremdschlüssel

```
CREATE TABLE institut (  
    nummer    INTEGER NOT NULL PRIMARY KEY,  
    vorname   VARCHAR(50),  
    name      VARCHAR(50),  
    instnr    INTEGER,  
    FOREIGN KEY(instnr) REFERENCES institut(nummer)  
);
```

# Tabellen anlegen (5)

---

verfügbare Datentypen:

- TINY-, MEDIUM-, SMALL-, BIGINT
- REAL / DOUBLE / FLOAT
- ENUM (Aufzählung)
- DATE / TIME / DATETIME
  
- CHAR / VARCHAR
- BINARY
  
- TEXT (MEDIUM-, LONG-)
- BLOB

# Mehr Möglichkeiten beim Anlegen einer Tabelle

---

- INDEX – Spalten, über die oft gesucht wird
- DEFAULT – falls beim Eintrag leer gelassen
- nach REFERENCES tabelle(feld)
  - ON DELETE ...
  - ON UPDATE ...
- AUTO\_INCREMENT – sinnvoll bei Schlüsseln
- Trigger

# Daten einfügen

---

- `INSERT INTO tabelle(feld1,feld2) VALUES(wert1,wert2) ;`
- `VALUES` kann mehrmals hintereinander stehen
- statt 'wert1' kann gar nichts stehen, wenn die Tabelle dies zulässt, oder auch `DEFAULT`
- Mögliche Fehler hier:
  - Einfügen von Werten, die Fremdschlüssel sein sollten, aber nicht sind
  - falsche Spaltennamen
  - keine Werte für Spalten, die als 'NOT NULL' angelegt wurden
- `REPLACE` anstatt `INSERT` überschreibt alte Werte mit gleichem Schlüssel

# Abfragen auf Tabellen

---

- einfach: `SELECT <feldnamen> FROM <tabelle>`
  - feldnamen: kommaseparierte Liste oder '\*' für alle
  - ergibt als Ergebnis ein Zeile pro Treffer aus
- Abfrage mit Bedingung: `SELECT <feldnamen> FROM <tabelle> WHERE <bedingung>`
  - bedingung:
    - `<feldname>="wert"`
    - `<feldname>=Zahl`
    - `<feldname> like "%ball%"` (trifft Uniball, Fussball, baldiun)
  - Verknüpfung von Bedingungen mit 'and' oder 'or'
  - Klammerungen sind möglich

# Abfragebedingungen formulieren

---

- nicht nur einfache Vergleiche mögliche, auch Summen, Minima, Maxima
- Vergleiche mit  $>$ ,  $<$  etc.
- Verwendung von MySQL-Funktionen, z.B.:
  - MIN, MAX, SUM
  - LENGTH, ASCII, LEFT, LOWER, UPPER
  - DATEDIFF
  - ...

- Gruppieren von Einträgen mit : `GROUP BY field`

- statt nur `SELECT` ggf. `SELECT DISTINCT`

- funktioniert:

```
SELECT MIN(plz) FROM institut;
```

- funktioniert nicht:

```
SELECT plz FROM institut WHERE plz=MIN(plz);
```

# Beispiele Abfragebedingungen (DISTINCT)

rechner	datum	stunden
merkur	2006-04-31	10
venus	2006-04-31	10
erde	2006-04-31	10
merkur	2006-04-30	20
venus	2006-04-30	20

> SELECT DISTINCT rechner FROM abrechnung;

```
+-----+
| rechner |
+-----+
| merkur  |
| venus   |
| erde    |
+-----+
```

# Beispiele Abfragebedingungen (GROUP BY, SUM)

rechner	datum	stunden
merkur	2006-04-31	10
venus	2006-04-31	10
erde	2006-04-31	10
merkur	2006-04-30	20
venus	2006-04-30	20

- `SELECT rechner, SUM(stunden) AS zeit FROM abrechnung GROUP BY rechner;`

```
+-----+-----+
| rechner | zeit |
+-----+-----+
| erde    | 10   |
| merkur  | 30   |
| venus   | 30   |
+-----+-----+
```

# Mehr Möglichkeiten

---

- HAVING • Bedingungen zu den Ergebnisspalten
- ORDER Sortieren nach Ergebnisspalten
- LIMIT Limitierung der Anzahl der Ergebnisspalten
- INTO OUTFILE ' ' " Export der Ergebnisse in eine Datei

- Reihenfolge muss eingehalten werden:

```
SELECT ... FROM ... [WHERE ...] [GROUP BY ...] [HAVING ... ]  
[ORDER ...] [LIMIT ...]
```

# Beispiel (HAVING)

rechner	datum	stunden
merkur	2006-04-31	10
venus	2006-04-31	10
erde	2006-04-31	10
merkur	2006-04-30	20
venus	2006-04-30	20

```
SELECT rechner,SUM(stunden) AS zeit FROM abrechnung
GROUP BY rechner HAVING zeit>20;
```

```
+-----+-----+
| rechner | zeit |
+-----+-----+
| merkur  |    30 |
| venus   |    30 |
+-----+-----+
```

# Beispiel (HAVING,ORDER)

rechner	datum	stunden
merkur	2006-04-31	10
venus	2006-04-31	10
erde	2006-04-31	10
merkur	2006-04-30	20
venus	2006-04-30	20

```
SELECT rechner,SUM(stunden) AS zeit FROM abrechnung  
GROUP BY rechner DESC HAVING zeit>20 ORDER BY rechner;
```

```
+-----+-----+  
| rechner | zeit |  
+-----+-----+  
| venus   |    30 |  
| merkur  |    30 |  
+-----+-----+
```

# Übersicht/Reihenfolge

---

```
SELECT [DISTINCT]
    SPALTE1 [AS ], SPALTE2 [AS ], ...
FROM
    TABELLE1, TABELLE2, ...
WHERE
    BEDINGUNG
GROUP BY
    SPALTE [ASC | DESC]
HAVING
    BEDINGUNG
ORDER BY
    FELD [ASC | DESC]
LIMIT
    Anzahl
```

# Abfragen über mehrere Tabellen (1)

---

- einfach: `SELECT <feldnamen> FROM <tabelle> WHERE <bedingung>`
- verschachtelt: Bedingung=eigene SQL-Abfrage
- Beispiel: ... `WHERE autor IN (SELECT name FROM personen  
WHERE arbeit="schriftsteller")`

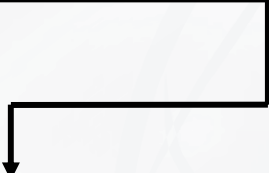
# Abfragen über mehrere Tabellen (2)

---

- Schlüsselwort `join`: Verbindung mehrerer Tabellen (über Fremdschlüssel)
- Syntax: `select ... from tabelleA join tabelle`
- `join` nur Datensätze, in denen alle gesuchten Werte in beiden Tabellen existieren
- `inner join` alle Datensätze aus linker und rechter Tabelle
- `left join` wie `join` + der Datensätze, die in der linken Tabelle stehen, aber in der rechten Tabelle keinen Eintrag haben
- `right join` der Datensätze, die in der rechten Tabelle stehen, aber in der linken Tabelle keinen Eintrag haben
  
- direkte Verbindung von Tabelle über Angabe von Schlüsseln

# Abfragen über mehrere Tabellen (3)

Nummer	Vorname	Name	Institutsnummer
1	Karl	Meier	1
2	Hans	Müller	3
3	Werner	Böhm	
...	...	...	...



Nummer	Name	Ort	PLZ
1	Chemie	Dresden	01146
2	Mathematik	Dresden	03423
3	Mathematik	München	
...	...	...	...

# Beispiel join

```
mysql>
```

```
select * from mitarbeiter join institut;
```

nummer	name	vorname	instnr	nummer	name	ort	plz
1	Karl	Meier	1	1	Chemie	Dresden	01146
2	Hans	Müller	3	1	Chemie	Dresden	01146
3	Werner	Böhm	NULL	1	Chemie	Dresden	01146
1	Karl	Meier	1	2	Mathematik	Dresden	03423
2	Hans	Müller	3	2	Mathematik	Dresden	03423
3	Werner	Böhm	NULL	2	Mathematik	Dresden	03423
1	Karl	Meier	1	3	Mathematik	München	NULL
2	Hans	Müller	3	3	Mathematik	München	NULL
3	Werner	Böhm	NULL	3	Mathematik	München	NULL

# Beispiel join on

---

mysql>

```
select * from mitarbeiter join institut on mitarbeiter.instnr=institut.nummer;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| nummer | name  | vorname | instnr | nummer | name      | ort      | plz  |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      1 | Karl  | Meier   |      1 |      1 | Chemie    | Dresden  | 01146 |
|      2 | Hans  | Müller  |      3 |      3 | Mathematik | München  | NULL  |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

# Beispiel left join on

---

mysql>

```
select * from mitarbeiter left join institut on mitarbeiter.instnr = institut.nummer;
```

nummer	name	vorname	instnr	nummer	name	ort	plz
1	Karl	Meier	1	1	Chemie	Dresden	01146
2	Hans	Müller	3	3	Mathematik	München	NULL
3	Werner	Böhm	NULL	NULL	NULL	NULL	NULL

# Beispiel right join on

---

mysql>

```
select * from mitarbeiter right join institut on mitarbeiter.instnr=institut.nummer;
```

nummer	name	vorname	instnr	nummer	name	ort	plz
1	Karl	Meier	1	1	Chemie	Dresden	01146
NULL	NULL	NULL	NULL	2	Mathematik	Dresden	03423
2	Hans	Müller	3	3	Mathematik	München	NULL

# Beispiel inner join on

---

mysql>

```
select * from mitarbeiter inner join institut on mitarbeiter.instnr=institut.nummer;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| nummer | name  | vorname | instnr | nummer | name      | ort      | plz  |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      1 | Karl  | Meier   |      1 |      1 | Chemie    | Dresden  | 01146 |
|      2 | Hans  | Müller  |      3 |      3 | Mathematik | München  | NULL  |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

# Aliase für Ergebnisse

---

- Problem: Tabellen mit den selben Spaltennamen
- Lösungen:
  - alle Spalten über alle Tabellen haben disjunkte Namen (z.B. tabelle\_feld)
  - Aliase fuer Ergebnisfelder
- Aliase im select-Teil eine Statements
- `SELECT feld AS alias, feld2 AS anderername from ...`
- Bei Namensgleichheit von Spalten und Abfragen über mehrere Tabellen:  
`SELECT mitarbeiter.name as person, institut.name as inst  
from ...`
- Aneinanderhängen von Spalten als eine Ergebnisspalte leistet: `CONCAT`  
`SELECT CONCAT(vorname, " ", name) as person from  
mitarbeiter;`

# Transaktionen

---

- entweder ganz oder gar nicht
- Markierung setzen, bevor eine Serie von Aktionen gestartet wird:
  - `START TRANSACTION`
- soll abgebrochen werden:
  - `ROLLBACK`
- alles bestätigen:
  - `COMMIT`
- Hinweis: `SET AUTOCOMMIT=0 | 1`

# Informationen über eine Datenbank gewinnen

---

## ● SHOW ...

- DATABASES
- TABLES
- ENGINES
- GRANTS
- OPEN TABLES
- LOGS
- PRIVILEGES
- STATUS

## ● DESCRIBE TABELLE tabelle

# Optimieren / Reparieren von Tabellen

---

- ANALYZE TABLE tabelle
- CHECK TABLE tabelle
- REPAIR TABLE tabelle
  
- BACKUP TABLE tabelle to 'dateiname'
- RESTORE TABLE tabelle from 'dateiname'

# Bestehende Tabelle verändern

---

- ALTER TABLE *tabelle* ...
  - ADD neuesfeld *DEFINITION* [FIRST | AFTER *bestehendesfeld* ]
  - ADD INDEX ...
  - ADD PRIMARY KEY ...
  - ADD FOREIGN KEY ...
  - ALTER feld SET DEFAULT ... | DROP DEFAULT
  - DROP PRIMARY/FOREIGN KEY ...
  - CHANGE altesfeld neuesfeld [DEFINITION]
  - ...

```
ALTER TABLE mitarbeiter CHANGE mail vorname VARCHAR(100);
```

# Backup / Dump

---

- Erlaubt Backups von allen MySQL-Daten
- wählbar, ob eine oder alle Datenbanken
- Syntax:
  - `mysqldump database >'backupfile'`
  - `mysqldump --databases db1 db2 db3 ... >'backupfile'`
  - `mysqldump --all-databases >'backupfile'`
- Format des Outputs: SQL-Kommandos
  
- Restore eines solchen Backups:
  - `mysql -u root -p <'backupfile'`
- danach unbedingt ein `flush privileges;` im mysql Client

# Löschen / Leeren von Tabellen

---

- Leeren einer bestehenden Tabelle

```
TRUNCATE TABLE name
```

- Löschen einer bestehenden Tabelle

```
DELETE TABLE name
```

- Löschen einer Datenbank

```
DROP DATABASE name
```

# Weitere Kommandozeilenprogramme:

---

- mysqlaccess - Testen von Zugriffsprivilegien
- mysqladmin - Testen von Funktionalitäten, auch Anlegen/Löschen von Datenbanken
- mysqlcheck - Testen/Reparieren von Tabellen
- mysqlimport - Daten aus einer Datei in ein File laden
- mysqlshow - Informationen zu Datenbanken und Tabellen

# Randbemerkungen

---

- intern verschiedene 'storage engines'

# Literatur

---

- Carsten Bormann: SQL, 2002, SPC TEIA Lehrbuch Verlag AG
- Paul DuBois: MySQL Kochbuch, O'Reilly Verlag, 2003