



DB: Datenbanken

SQL

Prof. Dr. Ludger Martin

Gliederung

- ★ Einführung
- ★ Zeilen einfügen
- ★ Abfrage
- ★ Zeilen ändern
- ★ Views

Einführung

★ Drei Ebenen des SQL-Standards

★ Kommandos wie `CREATE TABLE` oder `DROP TABLE` (DDL) ✓

★ Kommandos wie `INSERT`, `SELECT`, `UPDATE` oder `DELETE` (DML) **wird hier behandelt**

★ Programmiersprachen-Einbettung und -Anbindung

Einführung

★ Kommentare:

- ★ Vom Zeichen # bis zum Zeilenende
- ★ Von der Sequenz '--□' bis zum Zeilenende
- ★ Von der Sequenz /* bis zur folgenden Seq. */

Zeilen einfügen

```
★ INSERT [INTO] tbl_name  
  [ (col_name, ...) ]  
  VALUES (expr, ...) [,  
           (expr, ...) ] *
```

[] * und ...
gehören nicht zur
Syntax!

- ★ Spaltennamen optional, falls nicht angegeben gilt:
 - ★ Alle Werte müssen angegeben werden
 - ★ Reihenfolge in Tabelle muss beachtet werden
- ★ Wenn Spaltennamen angegeben, kann `AUTO_INCREMENT` genutzt werden, um Primärschlüsselfeld automatisch zu füllen
- ★ Zeichenreihen müssen in ' ' geschrieben werden

Zeilen einfügen

★ Beispiel:

```
INSERT INTO angestellte VALUES  
  ('1', 'Hans Fliege', 'Wiesbaden', 'Pilot'),  
  ('2', 'Fred Schraube', 'Bonn', 'Mechaniker');
```

★ `INSERT INTO pilot (angnr, std, liz) VALUES`
`('1', '150', 'Airbus 380');`

★ Beispiel AUTO_INCREMENT:

```
INSERT INTO angestellte  
  (name, adr, beruf)  
VALUES ('Paul Schmitt', 'Wiesbaden', 'Schweißer')
```

Der nächste eindeutige Wert für `angnr` wird automatisch gewählt

Anfragen

- ★ Komplexe Abfragemöglichkeiten mit `SELECT`

- ★ Einfachste Abfrage:

```
SELECT * FROM table_reference
```

- ★ Liest alle Spalten einer Tabelle (* steht für alle)

- ★ Liest alle Zeilen einer Tabelle

- ★ Beispiel:

```
SELECT * FROM angestellte
```

Anfragen

★ Auswahl der Spalten:

... gehört nicht
zur Syntax!

```
SELECT select_expr, ...  
FROM table_reference
```

Nach SELECT können Spalten durch Komma
getrennt angegeben werden

★ Beispiel:

```
SELECT name, adr FROM angestellte
```


Anfragen


- ★ `SELECT`-Kommandos liefern im Unterschied zu Relationen *Multimengen* (doppelte Tupel sind enthalten)
- ★ Durch Verwendung von `DISTINCT` können doppelte ausgeschlossen werden
- ★ Beispiel:

```
SELECT DISTINCT adr FROM angestellte
```

Anfragen

- ★ Bedingung, welche Daten ausgelesen:

```
SELECT select_expr, ...  
FROM table_reference  
WHERE where_condition
```



... gehört nicht zur Syntax!

- ★ Operatoren:

| Operator | Erklärung |
|----------|----------------|
| = | gleich |
| <> | ungleich |
| < | kleiner |
| > | größer |
| <= | kleiner gleich |
| >= | größer gleich |

Anfragen

★ Operatoren:

| Operator | Erklärung |
|----------|---------------------------|
| AND | logische UND-Verknüpfung |
| OR | logische ODER-Verknüpfung |
| NOT | Negierung einer Bedingung |

★ Mit () in Bedingungen kann eine Priorität angegeben werden

★ Mengenoperation:

IN (. . . , . . . , . . .)

... gehört nicht zur Syntax!

Anfragen

★ Beispiele:

- ★ Angestellte mit `adr` gleich 'Wiesbaden' suchen

```
SELECT * FROM angestellte  
WHERE adr='Wiesbaden'
```

- ★ Piloten suchen, die zwischen 100 und 200 Flugstunden haben

```
SELECT * FROM pilot  
WHERE (std>='100') AND (std<='200')
```

Anfragen

★ Ähnlichkeitsoperatoren:

★ `spalte LIKE '...'`

... gehört nicht zur Syntax!

★ Suche nach bestimmten Zeichenreihen in `spalte`

★ `%` gilt als Platzhalter für beliebig viele Zeichen

★ `_` gilt als Platzhalter für genau ein Zeichen

★ `spalte REGEXP muster`

★ Suche nach regulären Ausdruck `muster` in `spalte`

★ Posix-Syntax!

★ `spalte BETWEEN min AND max`

★ `spalte` im Wertebereich von `min` bis `max` (inkl. `min/max`)

Anfragen

★ Beispiele:

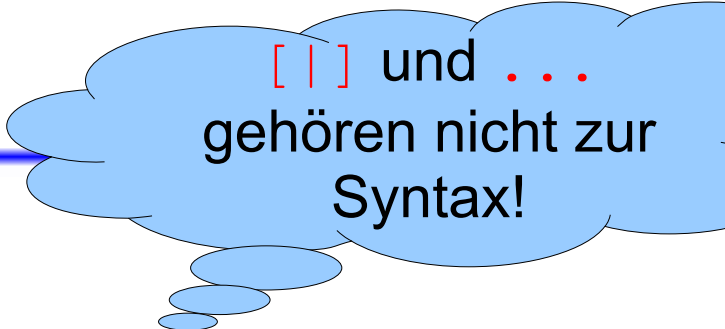
★ Angestellte mit Nachname *Fliege*

```
SELECT * FROM angestellte  
WHERE name LIKE '%Fliege%'
```

★ Piloten, die zwischen 100 und 200 Flugstunden haben

```
SELECT * FROM pilot  
WHERE std BETWEEN '100' AND '200'
```

Anfragen



[|] und ...
gehören nicht zur
Syntax!

★ Sortierung:

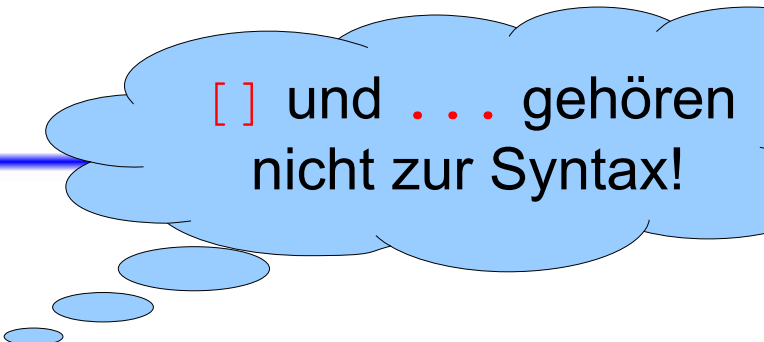
★ ORDER BY `col_name` [ASC | DESC] [, ...]

★ Sortierung aufsteigend oder absteigend nach einer oder mehreren Spalten

★ Beispiel:

```
SELECT * FROM angestellte  
ORDER BY name DESC
```

Anfragen



[] und ... gehören nicht zur Syntax!

★ Gruppierung

★ GROUP BY `col_name` [, ...]
[HAVING `where_condition`]

- ★ Datensätze mit gleichen Daten in einer Spalte werden zusammengefasst. D.h. nur eine Zeile wird angezeigt (Zufallsauswahl).
- ★ Mit HAVING wird erreicht, dass eine bestimmte Bedingung für die Gruppe gegeben sein muss

Anfragen

★ angestellte

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |
| 3 | Paul Schmitt | Wiesbaden | Schweißer |

★ `SELECT * FROM angestellte GROUP BY adr`

| an... | name | adr | beruf |
|-------|---------------|-----------|------------|
| 2 | Fred Schraube | Bonn | Mechaniker |
| 1 | Hans Fliege | Wiesbaden | Pilot |

Anfragen

★ Ausdrücke:

- ★ Anstelle der Angabe von reinen Spalten-Namen können auch Ausdrücke definiert werden
- ★ Numerische Operationen: + - * /
- ★ Numerische Funktionen:

| Funktion | Erklärung |
|----------|---------------------------|
| AVG | arithmetischer Mittelwert |
| MAX | höchster Wert |
| MIN | kleinster Wert |
| SUM | Summe |
| COUNT | Anzahl von Datensätzen |

Anfragen

★ angestellte

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |
| 3 | Paul Schmitt | Wiesbaden | Schweißer |

★ `SELECT *, count(*)`
`FROM angestellte`
`GROUP BY adr`

| an... | name | adr | beruf | count(*) |
|-------|---------------|-----------|------------|----------|
| 2 | Fred Schraube | Bonn | Mechaniker | 1 |
| 1 | Hans Fliege | Wiesbaden | Pilot | 2 |

Anfragen

★ angestellte

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |
| 3 | Paul Schmitt | Wiesbaden | Schweißer |

```
★ SELECT *, count(*)  
  FROM angestellte  
  GROUP BY adr  
  HAVING count(*) >=2
```

| an... | name | adr | beruf | count(*) |
|-------|-------------|-----------|-------|----------|
| 1 | Hans Fliege | Wiesbaden | Pilot | 2 |

Anfragen

★ Umbenennen von Spalten:

★ `col_name AS new_name`

★ Ein anderer Name kann für ein Abfrage-Ergebnis festgelegt werden

★ Beispiel:

```
SELECT adr, count(*) AS anzahl  
FROM angestellte  
GROUP BY adr
```

Anfragen

Verbund-Operatoren

- ☆ In einer `SELECT`-Anweisung können mehrere Tabellen miteinander verknüpft werden
- ☆ Angabe mehrerer Tabellennamen nach `FROM`
 - ☆

```
SELECT select_expr, ...  
      FROM table_reference,  
           table_reference, ...  
      WHERE where_condition
```
 - ☆ Verknüpfung geschieht in `WHERE` Bedingung
 - ☆ Spaltenreferenzierung mit Tabellename und `.` oder mit Tabellen-Alias
 - ☆ Wird unübersichtlich, wenn umfangreiche Bedingung notwendig

Anfragen

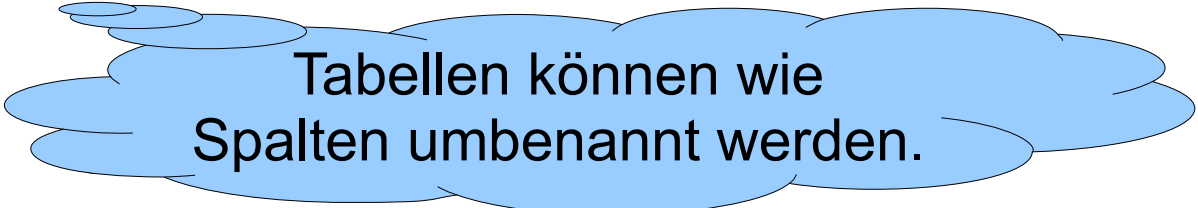
Verbund-Operatoren

★ Beispiel Verknüpfung:

```
SELECT angestellte.name,  
       pilot.std  
FROM angestellte, pilot  
WHERE angestellte.angnr =  
       pilot.angnr
```

oder

```
SELECT a.name, p.std  
FROM angestellte AS a, pilot AS p  
WHERE a.angnr = p.angnr
```



Tabellen können wie
Spalten umbenannt werden.

Anfragen

Verbund-Operatoren

★ Nutzung von JOIN

`{ } [] | . . .` gehören
nicht zur Syntax!

```
SELECT select_expr, . . .  
FROM table_reference  
    { NATURAL JOIN join_table_ref. }  
    | {{ [INNER]  
      | {{LEFT|RIGHT} [OUTER]}  
      } JOIN join_table_ref.  
    ON join_condition }  
WHERE where_condition
```


Anfragen

Verbund-Operatoren

★ NATURAL JOIN:

- ★ Verbindet zwei Tabellen anhand von Spalten mit gleichen Namen
- ★ Es wird immer nur eine Spalte mit gleichem Namen ausgegeben
- ★ Gibt nur die Datenreihen aus, welche gemeinsame Übereinstimmung in den Tabellen besitzen (Nur Angestellte, die auch Piloten sind)

Anfragen

Verbund-Operatoren

★ angestellte

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |
| 3 | Paul Schmitt | Wiesbaden | Schweißer |

pilot

| angnr | std | liz |
|-------|-----|------------|
| 1 | 150 | Airbus 380 |

★ SELECT *

FROM angestellte

NATURAL JOIN pilot

| angnr | name | adr | beruf | std | liz |
|-------|-------------|-----------|-------|-----|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot | 150 | Airbus 380 |

angnr kommt im
Ergebnis nur 1x vor

Anfragen

Verbund-Operatoren

★ INNER JOIN:

- ★ Gibt nur die Datenreihen aus, welche eine gemeinsame Übereinstimmung in den Tabellen besitzen (Nur Angestellte, die auch Piloten sind)
- ★ Nach ON wird Verknüpfungsbedingung angegeben, unabhängig von WHERE Bedingung

Anfragen

Verbund-Operatoren

★ angestellte

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |
| 3 | Paul Schmitt | Wiesbaden | Schweißer |

pilot

| angnr | std | liz |
|-------|-----|------------|
| 1 | 150 | Airbus 380 |

★ SELECT *

```
FROM angestellte AS a
  INNER JOIN pilot AS p
    ON a.angnr = p.angnr
```

| angnr | name | adr | beruf | angnr | std | liz |
|-------|-------------|-----------|-------|-------|-----|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot | 1 | 150 | Airbus 380 |

angnr kommt im
Ergebnis nur 2x vor

Anfragen

Verbund-Operatoren

★ OUTER JOIN:

- ★ Beinhaltet auch Datenreihen, die nicht in beiden Tabellen enthalten sind
- ★ Mit `LEFT` und `RIGHT` kann die Tabelle spezifiziert werden, die vollständig enthalten sein soll
- ★ `LEFT` und `RIGHT` bezieht sich auf die Tabellen angrenzend an `JOIN`
- ★ Nicht belegte Felder werden mit `NULL` gefüllt

Anfragen

Verbund-Operatoren

★ angestellte

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |
| 3 | Paul Schmitt | Wiesbaden | Schweißer |

pilot

| angnr | std | liz |
|-------|-----|------------|
| 1 | 150 | Airbus 380 |

★ SELECT *

```
FROM angestellte AS a
  RIGHT OUTER JOIN pilot AS p
    ON a.angnr = p.angnr
```

| angnr | name | adr | beruf | angnr | std | liz |
|-------|-------------|-----------|-------|-------|-----|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot | 1 | 150 | Airbus 380 |

Anfragen

Verbund-Operatoren

★ angestellte

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |
| 3 | Paul Schmitt | Wiesbaden | Schweißer |

pilot

| angnr | std | liz |
|-------|-----|------------|
| 1 | 150 | Airbus 380 |

★ SELECT *

```
FROM angestellte AS a
LEFT OUTER JOIN pilot AS p
ON a.angnr = p.angnr
```

| angnr | name | adr | beruf | angnr | std | liz |
|-------|---------------|-----------|------------|--------|--------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot | 1 | 150 | Airbus 380 |
| 2 | Fred Schraube | Bonn | Mechaniker | <null> | <null> | <null> |
| 3 | Paul Schmitt | Wiesbaden | Schweißer | <null> | <null> | <null> |

Anfragen

Verbund-Operatoren

★ UNION

- ★ Ergebnisse einer Auswahl von Anweisungen zu einer Ergebnismenge zusammenfassen
- ★ Anzahl und Typ der ausgewählten Spalten müssen identisch sein
- ★ ORDER usw. muss bzw. kann am Ende stehen

★ SELECT . . .
UNION [ALL | DISTINCT] SELECT . . .
[ORDER BY . . .]

[|] und . . .
gehören nicht zur
Syntax!

Anfragen

Verbund-Operatoren

★ angestellte

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |
| 3 | Paul Schmitt | Wiesbaden | Schweißer |

```
★ SELECT * FROM angestellte
    WHERE beruf='Pilot'
UNION
    SELECT * FROM angestellte
    WHERE beruf='Mechaniker'
```

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |

Anfragen

Verbund-Operatoren

★ Unterabfragen:

- ★ SELECT Anweisung innerhalb einer anderen SELECT Anweisung
- ★ Strukturierte Abfragen
- ★ Alternative Möglichkeit zu komplexen JOIN oder UNION Abfragen, sollen lesbarer sein
- ★ SELECT *
FROM `tr1`
WHERE `column1` =
(SELECT `column1` FROM `tr2`)

Anfragen

Verbund-Operatoren

★ Zwei mögliche Ergebnisse von Unterabfragen zulässig

- ★ Ein einzelnes Ergebnis wird von `SELECT` zurück geliefert:


```
... column1 =  
    (SELECT max(column2) FROM tr2)
```

- ★ Ein n-Tupel wird vom `SELECT` zurück geliefert

- ★ Vergleich mit Mengenoperation `IN (...)`
äquivalent zu `= ANY (...)`

- ★ Benutzung der Schlüsselworte

```
column op ALL | ANY (...)  
s1 > ALL (...)
```



| und ... gehören
nicht zur Syntax!

Anfragen

Verbund-Operatoren

★ angestellte

| angnr | name | adr | beruf |
|-------|---------------|-----------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot |
| 2 | Fred Schraube | Bonn | Mechaniker |
| 3 | Paul Schmitt | Wiesbaden | Schweißer |

pilot

| angnr | std | liz |
|-------|-----|------------|
| 1 | 150 | Airbus 380 |

★ `SELECT * FROM angestellte
WHERE angnr IN (SELECT angnr
FROM pilot
WHERE std>100)`

| angnr | name | adr | beruf |
|-------|-------------|-----------|-------|
| 1 | Hans Fliege | Wiesbaden | Pilot |

Anfragen

★ Ausdrücke:

★ Zeichenreihen-Funktionen:

★ SUBSTRING (str, pos, len)

★ CHAR_LENGTH (str)

★ CONCAT (str1, str2, ...)

★ Datum-Funktionen:

★ CURDATE ()

★ CURTIME ()

★ Noch viel mehr Funktionen

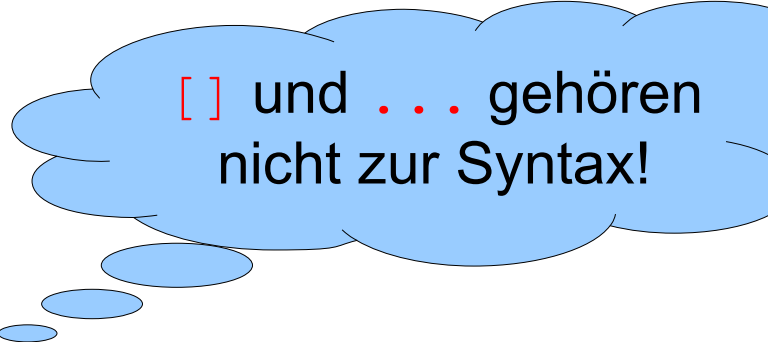
➡ <http://dev.mysql.com/doc>

Zeilen ändern

- ★ Bereits bestehende Zeilen können geändert werden

- ★

```
UPDATE table_reference  
    SET col_name1=expr1[,  
        ...]  
    [WHERE where_condition]
```



[] und ... gehören nicht zur Syntax!

- ★ WHERE **Bedingung** wie bei SELECT

- ★ **Beispiel:**

```
UPDATE pilot SET std='200'  
    WHERE angnr='1'
```

Zeilen ändern

[,] und ...
gehören nicht zur
Syntax!

- ★ Eine, mehrere oder alle Zeilen löschen

- ★ DELETE FROM `table_reference`
[WHERE `where_condition`]

- ★ WHERE **Bedingung** wie bei SELECT

- ★ DELETE kann auch für Zeilen in mehreren
Tabellen genutzt werden

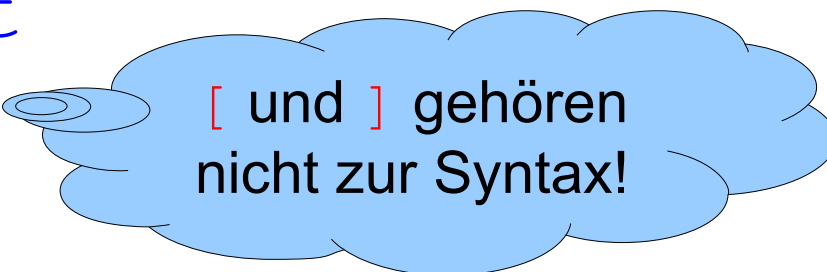
```
DELETE tbl_name[ , tbl_name] ...  
FROM table_references  
[WHERE where_condition]
```

auch JOIN

Views

- ★ Spezielle Benutzersicht
- ★ „virtuelle“ Tabelle
- ★ Werden erst zur Laufzeit ausgewertet
- ★ In einem View können auch mit `UPDATE` Daten aktualisiert werden

★ `CREATE VIEW view_name`
 `[(column_list)]`
 `AS select_statement`
 `[WITH CHECK OPTION]`



`[und]` gehören
nicht zur Syntax!

Views

- ★ Spezielle Benutzersicht
- ★ „virtuelle“ Tabelle
- ★ Werden erst zur Laufzeit ausgewertet
- ★ In einem View können auch mit UPDATE Daten aktualisiert werden

★ `CREATE VIEW view_name`
 `[(column_list)]`
 `AS select_statement`
 `[WITH CHECK OPTION]`

Die Anzahl der Einträge in `column_list` muss identisch mit den aus `select_statement` sein!

Views

- ★ Spezielle Benutzersicht
- ★ „virtuelle“ Tabelle
- ★ Werden erst zur Laufzeit ausgewertet
- ★ In einem View können auch mit UPDATE Daten aktualisiert werden

★ `CREATE VIEW view_name`
 `[(column_list)]`
 `AS select_statement`
 `[WITH CHECK OPTION]`

WITH CHECK OPTION lässt
das UPDATE nur zu, wenn die
WHERE Bedingung auf
`select_statement` passt!

View

★ Beispiel:

```
CREATE VIEW v_pilot AS
  SELECT a.*, p.std, p.liz
  FROM angestellte AS a
  LEFT OUTER JOIN pilot AS p
  ON a.angnr = p.angnr;
```

```
SELECT * FROM v_pilot;
```

| angnr | name | adr | beruf | std | liz |
|-------|---------------|-----------|------------|--------|------------|
| 1 | Hans Fliege | Wiesbaden | Pilot | 200 | Airbus 380 |
| 2 | Fred Schraube | Bonn | Mechaniker | <null> | <null> |
| 3 | Paul Schmitt | Wiesbaden | Schweißer | <null> | <null> |

Literatur

- ★ Pröll, S., Zangerle, E. und Gassler, W.: MySQL – Das umfassende Handbuch, 2. Auflage, Galileo Press, 2013
- ★ Vossen, Gottfried: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, 5. Auflage, Oldenburg Wissenschaftsverlag, 2008
- ★ Lubkowitz, M: Webseiten programmieren und gestalten, Galileo Press, 2004
- ★ Sun Microsystems: MySQL 5.1 Reference Manual, <http://dev.mysql.com/doc/refman/5.1/en/>