

# Datenbanksysteme 2011

noch Kapitel 7:  
SQL

Vorlesung vom 17.05.2011

Oliver Vornberger

Institut für Informatik  
Universität Osnabrück

[Sprung Selfjoin](#)

# SQL

- 1970 Edgar Codd:  
*A relational model for large shared data banks*
- 1975 SEQUEL für System R von IBM
- 1977 Oracle gegründet
- 1979 SQL
- 1992 SQL-2, SQL-92, SQL:1992
- 1999 SQL-3, SQL-99, SQL:1999 (objektorientiert)
- 2003 SQL:2003 (XML)
- 2006 SQL:2006 (XQuery)
- 2008 SQL:2008 (Merge, instead of triggers, ...)

# Relationale Datenbanksysteme

- DB2 IBM
- Informix IBM
- Database 11g Oracle
- Access Microsoft
- SQL Server Microsoft
- Ingres Open Source
- Postgres Open Source
- MySQL Open Source

# MySQL

Populärstes OpenSource Datenbanksystem

Verfügbar für Linux, Windows, Mac OS X

70.000 Downloads am Tag, > 10.000.000 Installationen

1994 entstanden als Version 3.21 aus mSQL  
von Michael Widenius, MySQL AB

2005 Version 5 (mit View, Trigger, Stored Procedures)

2008 Sun Microsystems kauft MySQL (1 Milliarde US-\$)

2010 Oracle kauft Sun Microsystems (7 Milliarden US-\$)

aktuelle Version: 5.5.11

# LAMP

- **L**inux
- **A**pache
- **M**ySQL
- **P**HP

# phpmyadmin

The screenshot shows the phpMyAdmin interface in a Mozilla Firefox browser window. The address bar shows the URL: <http://dbs.informatik.uos.de/phpmyadmin/index.php?db=UniWeb&lang=de-utf-8&c>. The browser title is "phpMyAdmin 2.11.8.1deb5 - Mozilla Firefox".

The interface displays the following information:

- Server: localhost
- Datenbank: UniWeb
- Tabelle: Studenten

Navigation buttons include: Anzeigen, Struktur, SQL, Suche, Einfügen, Exportieren, Importieren, Operationen, Leeren, and Löschen.

A message box indicates: "Zeige Datensätze 0 - 13 (14 insgesamt, die Abfrage dauerte 0.0001 sek.)".

The SQL-Befehl section shows the following query:

```
SELECT Name, Titel
FROM Studenten, hoeren, Vorlesungen
WHERE Studenten MatrNr = hoeren MatrNr
AND hoeren VorlNr = Vorlesungen VorlNr
LIMIT 0, 30
```

Below the query, there are options to "Messen", "Bearbeiten", "SQL erklären", "PHP-Code erzeugen", and "Aktualisieren".

The display settings show: "Zeige: 30 Datensätze, beginnend ab 0".

The sorting options are: "untereinander" and "angeordnet und wiederhole die Kopfzeilen nach 100 Datensätzen".

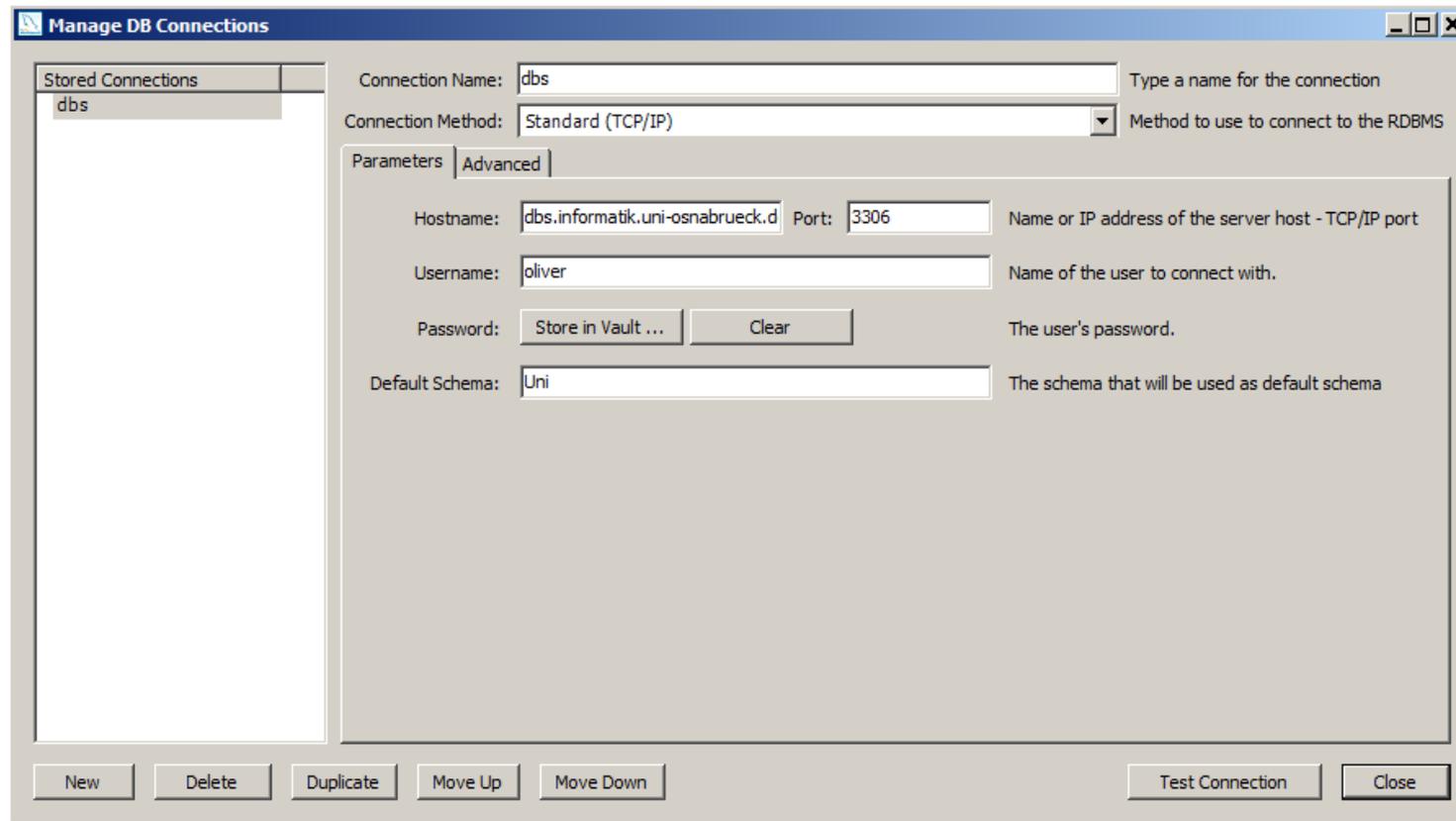
The results table is as follows:

| Name         | Titel             |
|--------------|-------------------|
| Jonas        | Glaube und Wissen |
| Fichte       | Grundzuege        |
| Schopenhauer | Logik             |
| Schopenhauer | Die 3 Kritiken    |

The status bar at the bottom left shows "Fertig".

<http://dbs.informatik.uni-osnabrueck.de/phpmyadmin>

# MySQL Workbench: manage connections



# MySQL WorkBench: SQL

The screenshot displays the MySQL Workbench interface. The main window is titled "SQL Editor (dbs)" and contains a menu bar (File, Edit, View, Query, Database, Plugins, Scripting, Community, Help) and a toolbar. The left sidebar shows the "Object Browser" with "ACTIONS" (Execute SQL File, Add Schema, Add Table, Add View, Add Routine) and "SCHEMAS" (Uni, UniTest, UniWeb). The "Object Information" panel at the bottom left is empty.

The central "Query 1" editor contains the following SQL query:

```
1 select titel, sws
2 from Professoren, Vorlesungen
3 where persnr=gelesenvon
4 and name='Sokrates'
```

The "Query 1 Result" panel shows the execution output. It includes a toolbar and a status bar indicating "Fetched 3 records. Duration: 0.000 sec, fetched in: 0.000 sec". The results are displayed in a table:

| titel   | sws |
|---------|-----|
| Logik   | 4   |
| Ethik   | 4   |
| Mäeutik | 2   |

The status bar at the bottom of the window indicates "Active schema changed to Uni".

# MySQL Referenzhandbuch

The screenshot shows a Mozilla Firefox browser window displaying the MySQL 5.1 German reference manual. The page title is "MySQL 5.1 Referenzhandbuch". The main content area features the heading "MySQL 5.1 Referenzhandbuch" and a paragraph explaining that this is a German translation of the MySQL reference manual, which is available in English on [dev.mysql.com](http://dev.mysql.com). It notes that the German version covers MySQL up to version 5.1. Below this, there is a copyright notice for 1997-2008 MySQL AB and Sun Microsystems, Inc., followed by a disclaimer stating that the documentation is not under the GPL license and that its use is restricted to private purposes.

On the left side, there is a "Documentation Library" section with a "Table of Contents" and a "Search manual:" input field. Below that, there are "Additional languages" listed: Chinese, English, and Japanese. A "Section Navigation" sidebar on the right lists 15 sections, including "Vorwort", "Allgemeine Informationen über MySQL", "Installation von MySQL", and "Erstellung einer eigenen".

<http://dev.mysql.com/doc/refman/5.1/de/>

# SQL: numerische Datentypen

|   |                     |   |
|---|---------------------|---|
| 8 | <b>bigint</b>       | ganze Zahlen von $-2^{63}$ bis $+2^{63}$  |
| 4 | <b>int</b>          | ganze Zahlen von $-2^{31}$ bis $+2^{31}$  |
| 3 | <b>mediumint</b>    | ganze Zahlen von $-2^{23}$ bis $+2^{23}$  |
| 2 | <b>smallint</b>     | ganze Zahlen von $-2^{15}$ bis $+2^{15}$  |
| 1 | <b>tinyint</b>      | ganze Zahlen von -128 bis +127  |
| 1 | <b>bit</b>          | ganze Zahlen von 0 bis 1  |
| 1 | <b>boolean</b>      | alias für tinyint   |
| d | <b>decimal(n,k)</b> | feste Genauigkeit, n Stellen, davon k nach Komma<br>9 digits in 4 Bytes, d.h. $d \approx \lceil n/9 \rceil * 4$ |
| d | <b>numeric(n,k)</b> | alias für decimal   |
| 4 | <b>float</b>        | Gleitkommazahlen von $-10^{38}$ bis $+10^{38}$  |
| 8 | <b>double, real</b> | Gleitkommazahlen von $-10^{308}$ bis $+10^{308}$  |

Microsoft SQL Server:

|   |              |   |
|---|--------------|---|
| 8 | <b>money</b> | Währungswerte mit 4 Nachkommastellen (MS SQL) |
|---|--------------|---|

# SQL Datentypen für Zeitangaben

- |   |                  |   |
|---|------------------|---|
| 1 | <b>year</b>      | von 1901 to 2155  |
| 3 | <b>date</b>      | von 01.01.0001 bis 31.12.9999<br>kodiert als $DD + 32*MM + 32*16*YYYY$  |
| 8 | <b>datetime</b>  | von 00.00.0000 00:00:00<br>bis 31.12.9999 23:59:59<br>kodiert als $YYYY*10000 + MM*100 + DD$<br>$HH*10000 + MM*100 + SS$<br>(nicht als Millisekunden seit 1.1.1970) |
| 4 | <b>timestamp</b> | von 01.01.1970 bis 31.12.2037<br>(beim Einfügen, inkl. Uhrzeit)<br>kodiert als Sekunden nach 1.1.1970   |
| 3 | <b>time</b>      | von -838:59:59 bis 838:59:59<br>kodiert als $HH*3600 + MM*60 + SS$  |

# SQL: Datentypen für Zeichenketten

**n**     **char(n)**     Zeichenkette fester Länge  
mit  $n \leq 255$  Zeichen

**n+d**   **varchar(n)** Zeichenkette variabler Länge  
mit  $n \leq 65535$  Zeichen  
[zusätzlich d Bytes für Längenangabe]

**n+d**   **text**     Zeichenkette variabler Länge  
[zusätzlich d Bytes für Längenangabe]

die ersten 256 Zeichen in Originaltabelle  
die nächsten Zeichen in 2000-Bytes-Blöcken  
in verborgenen Tabellen

# SQL: Datentypen für Binärdaten

n     **binary(n)**     Binärdaten fester Länge  
mit  $n \leq 255$  Bytes

n+d   **varbinary(n)**   Binärdaten variabler Länge  
mit  $n \leq 65535$  Bytes  
[zusätzlich d Bytes für Längenangabe]

n+x   **blob**            Binärkette variabler Länge  
[zusätzlich x Bytes für Verwaltung]

die ersten 256 Bytes in Originaltabelle  
die nächsten Zeichen in 2000-Bytes-Blöcken  
in verborgenen Tabellen

# SQL Mengen und Aufzählungen

- 8 `set` Menge von bis zu 64 Elementen
- 2 `enum` Liste von bis zu 65.535 Elementen

# SQL: create

```
Create table Personen (  
    persnr          int primary key auto_increment,  
    name            char(30) not null,  
    geschlecht      boolean default 0,  
    note            decimal (3,2),  
    groesse         float,  
    gewicht         double,  
    gebDatum        date,  
    einschulung     year,  
    marathon        time,  
    bemerkung       text,  
    photo           blob,  
    zugriff         timestamp,  
    kombination     set ('rot ', 'gruen', 'blau')  
    ) auto_increment = 100000;
```

# SQL: alter, modify, drop

Tabelle um eine Spalte erweitern:

```
alter table Personen  
add Vorname varchar(15)
```

Tabellenspalte ändern:

```
alter table Personen  
modify Vorname varchar(20)
```

Tabelle um eine Spalte verkürzen:

```
alter table Personen  
drop column Vorname
```

Tabelle entfernen:

```
drop table Personen
```

# SQL: Schlüsselworte

|          |          |        |
|----------|----------|--------|
| select   | distinct | in     |
| from     | count    | not    |
| where    | sum      | null   |
| order by | avg      | exists |
| asc      | max      | all    |
| desc     | min      | some   |
| as       | group by |        |
| like     | having   |        |
| upper    |          |        |
| lower    |          |        |

[zum Quiz Kant](#)

# SQL: select, from, where

1.) Liste alle Studenten:

```
select * from Studenten
```

2.) Liste Personalnummer und Name der C4-Professoren:

```
select PersNr, Name  
from Professoren  
where Rang='C4'
```

# SQL: count, as, is not, null

3.) Zähle alle Studenten

```
select count(*) from Studenten
```

4.) Liste Name und Studiendauer in Jahren von allen Studenten:

```
select Name, Semester/2 as Studienjahr  
from Studenten  
where Semester is not null
```

# SQL: between, in

5.) Liste alle Studenten mit Semesterzahlen zwischen 1 und 4:

```
select *  
from Studenten  
where Semester >= 1 and Semester <= 4
```

alternativ

```
select *  
from Studenten  
where Semester between 1 and 4
```

alternativ

```
select *  
from Studenten  
where Semester in (1,2,3,4)
```

# SQL: like, order, distinct

6.) Liste alle Vorlesungen mit `Ethik` im Titel:

```
select * from Vorlesungen
where Titel like '%ETHIK'
```

7.) Liste Personalnummer, Name und Rang aller Professoren, absteigend sortiert nach Rang, innerhalb des Rangs aufsteigend sortiert nach Name:

```
select PersNr, Name, Rang
from Professoren
order by Rang desc, Name asc
```

8.) Liste alle verschiedenen Ränge der Relation Professoren:

```
select distinct Rang
from Professoren
```

# SQL: Datum

9.) Liste alle Geburtstage mit ausgeschriebenem Monatsnamen:

```
select name,  
       Day    (Gebdatum) as Tag,  
       Month  (GebDatum) as Monat,  
       Year   (GebDatum) as Jahr  
from Studenten
```

10.) Liste das Alter der Studenten in Jahren:

```
select name, year(Now())-year(gebdatum) as Jahre  
from Studenten
```

# SQL Datumsfunktionen

11.) Liste die Wochentage der Geburtsdaten der Studenten:

```
select name,  
dayname(GebDatum) as Wochentag  
from Studenten
```

12.) Liste die Kalenderwochen der Geburtsdaten der Studenten:

```
select name,  
week(GebDatum) as Kalenderwoche  
from Studenten
```

# SQL: Verbund

13.) Liste den Dozenten der Vorlesung Logik:

```
select  Name, Titel
from    Professoren, Vorlesungen
where   PersNr = gelesenVon and Titel = 'Logik'
```

14.) Liste die Namen der Studenten mit ihren Vorlesungstiteln:

```
select  Name, Titel
from    Studenten, hoeren, Vorlesungen
where   Studenten.MatrNr = hoeren.MatrNr
and     hoeren.VorlNr    = Vorlesungen.VorlNr
```

alternativ:

```
select  s.Name, v.Titel
from    Studenten s, hoeren h, Vorlesungen v
where   s.MatrNr = h.MatrNr
and     h.VorlNr = v.VorlNr
```

# SQL: Self Join

- 15.) Liste die Namen der Assistenten, die für denselben Professor arbeiten, für den Aristoteles arbeitet:

```
select  a2.Name
from    Assistenten a1, Assistenten a2
where   a2.boss    =  a1.boss
and     a1.name    =  'Aristoteles'
and     a2.name    != 'Aristoteles'
```

# Self Join

Wer ist älter als Kant ?

```
select p1.name  
from Professoren p1, Professoren p2  
where p2.name = 'Kant'  
and p1.gebdatum < p2.gebdatum
```

# SQL: avg, group

16.) Liste die durchschnittliche Semesterzahl:

```
select  avg(Semester)
from    Studenten
```

17.) Liste Geburtstage der Gehaltsklassenältesten (ohne Namen !):

```
select  Rang, min(GebDatum) as Ältester
from    Professoren
group by Rang
```

18.) Liste Summe der SWS pro Professor:

```
select gelesenVon as PersNr, sum(SWS) as Lehrbelastung
from    Vorlesungen
group by gelesenVon
```

# SQL: having

- 19.) Liste Summe der SWS pro Professor, sofern seine Durchschnitts-SWS größer als 3 ist:

```
select gelesenVon as PersNr, sum(SWS) as  
Lehrbelastung  
from Vorlesungen  
group by gelesenVon  
having avg(SWS) > 3
```

# SQL: where & having

20.) Liste Summe der SWS pro C4-Professor, sofern seine Durchschnitts-SWS größer als 3 ist:

```
select  Name, sum(SWS)
from    Vorlesungen, Professoren
where   gelesenVon = PersNr and Rang='C4'
group  by gelesenVon, Name
having  avg(SWS) > 3.0
```

# SQL: Sub-Query

21.) Liste alle Prüfungen mit der schlechtesten Note:

```
select *  
from pruefen  
where Note = (select max(Note) from pruefen)
```

22.) Liste alle Professoren zusammen mit ihrer Lehrbelastung:

```
select PersNr,  
       Name,  
       (select sum(SWS)  
        from Vorlesungen  
        where gelesenVon = PersNr) as Lehrbelastung  
from Professoren
```

# SQL: exists

23.) Liste Studenten, die älter sind als der jüngste Professor:

```
select s.*
from Studenten s
where exists (select *
              from Professoren p
              where p.GebDatum > s.GebDatum)
```

alternativ:

```
select s.*
from Studenten s
where s.GebDatum < (select max(p.GebDatum)
                   from Professoren p )
```

## SQL: Verbund mit <

24.) Liste alle Assistenten, die jünger sind als ihr Professor

```
select  a.name, a.gebdatum, p.name, p.gebdatum
from    Assistenten a, Professoren p
where   a.boss = p.persNr
and     a.gebDatum < p.gebDatum
```

# SQL: geschachtelt

25.) Liste alle Studenten mit der Zahl ihrer Vorlesungen, sofern diese Zahl größer als 2 ist:

```
select tmp.MatrNr, tmp.Name, tmp.VorlAnzahl
from (select s.MatrNr,s.Name,count(*) as VorlAnzahl
      from Studenten s, hoeren h
      where s.MatrNr = h.MatrNr
      group by s.MatrNr) tmp
where tmp.VorlAnzahl > 2
```

alternativ:

```
select s.MatrNr, s.Name, count(*) as VorlAnzahl
from Studenten s, hoeren h
where s.MatrNr = h.MatrNr
group by s.MatrNr
having count(*) > 2
```

# SQL: geschachtelt

26.) Liste die Namen und Geburtstage der Gehaltsklassenältesten:

```
select name, rang, gebdatum
from Professoren
where (rang, gebdatum) in
      (select rang, min(gebdatum)
       from Professoren
       group by Rang)
```

27.) Liste Vorlesungen zusammen mit Marktanteil  
definiert als Hörerzahl/Gesamtzahl:

```
select h.VorlNr, h.AnzProVorl, g.GesamtAnz,
h.AnzProVorl / g.GesamtAnz as Marktanteil
from (select VorlNr, count(*) as AnzProVorl
      from hoeren group by VorlNr) h,
      (select count(*) as GesamtAnz
       from Studenten) g
```

# SQL: union, minus, intersect

28.) Liste Vereinigung von Professoren- und Assistenten-Namen:

```
(select Name from Assistenten)
```

```
union
```

```
(select Name from Professoren)
```

29.) Liste die Differenz von Professoren- und Assistenten-Namen (nur SQL-92):

```
(select Name from Assistenten)
```

```
minus
```

```
(select Name from Professoren)
```

30.) Liste den Durchschnitt von Professoren- und Assistenten-Namen (nur SQL-92):

```
(select Name from Assistenten)
```

```
intersect
```

```
(select Name from Professoren)
```

# SQL: not, in, exists

31.) Liste alle Professoren, die keine Vorlesung halten:

```
select Name
from Professoren
where PersNr not in ( select gelesenVon
                      from Vorlesungen )
```

alternativ:

```
select Name
from Professoren
where not exists ( select *
                  from Vorlesungen
                  where gelesenVon = PersNr )
```

# SQL: all, some

32.) Liste Studenten mit größter Semesterzahl:

```
select Name
from Studenten
where Semester >= all ( select Semester
                        from Studenten )
```

33.) Liste Studenten, die nicht die größte Semesterzahl haben:

```
select Name
from Studenten
where Semester < some ( select Semester
                       from Studenten )
```

# SQL: not, exists

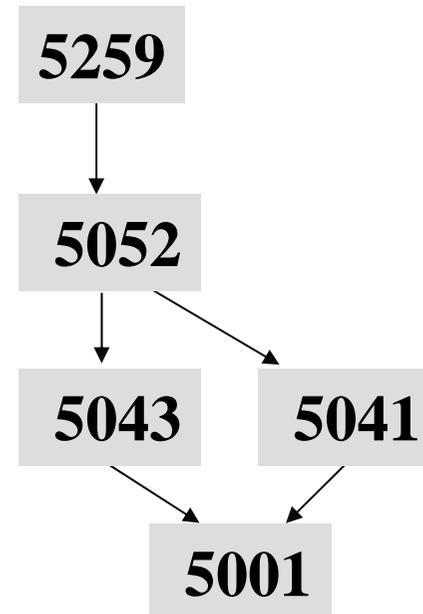
34.) Liste Studenten, die alle 4-stündigen Vorlesungen hören:

```
select s.*
from Studenten s
where not exists
    (select *
     from Vorlesungen v
     where v.SWS = 4 and not exists
         (select *
          from hoeren h
          where h.VorlNr = v.VorlNr
            and h.MatrNr = s.MatrNr
         )
    )
)
```

# Transitive Hülle

35.) Liste alle Voraussetzungen für die Vorlesung "Der Wiener Kreis"

| vorgaenger | nachfolger |
|------------|------------|
| 5001       | 5041       |
| 5001       | 5043       |
| 5001       | 5049       |
| 5041       | 5052       |
| 5043       | 5052       |
| 5041       | 5216       |
| 5052       | 5259       |



⇒ Transitive Hülle einer rekursiven Relation

# Prolog: Transitiv Hülle

**Trans(V,N) :- voraussetzen(V,N).**

**Trans(V,N) :- Trans(V,Z), voraussetzen(Z,N)**

# DB2: Transitive Hülle

```
with Trans(vorgaenger, nachfolger)
as (select vorgaenger, nachfolger from voraussetzen
    union all
    select t.vorgaenger, v.nachfolger
    from Trans t, voraussetzen v
    where t.nachfolger = v.vorgaenger)

select titel from Vorlesungen where vorlnr in
(select vorgaenger from Trans where nachfolger in
(select vorlnr from Vorlesungen
where titel='Der Wiener Kreis'))
```