Datenbanksysteme 2011

Kapitel 7:
SQL

Oliver Vornberger

Institut für Informatik
Universität Osnabrück

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 Microsystem kauft MySQL (1 Milliarde US-$)

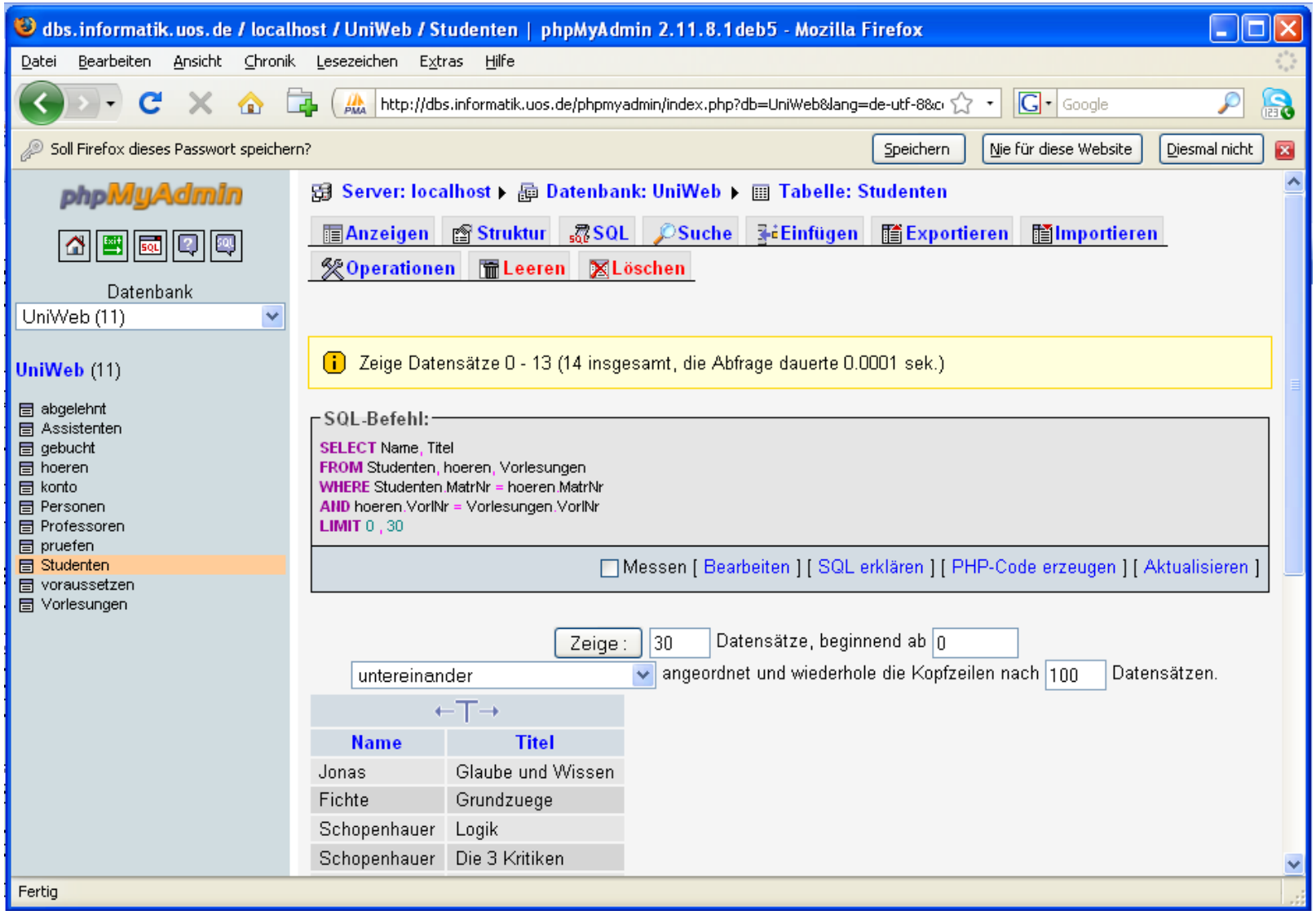
2010 Oracle kauft Sun Microsystem (7 Milliarden US-$)

aktuelle Version: 5.5.11

LAMP

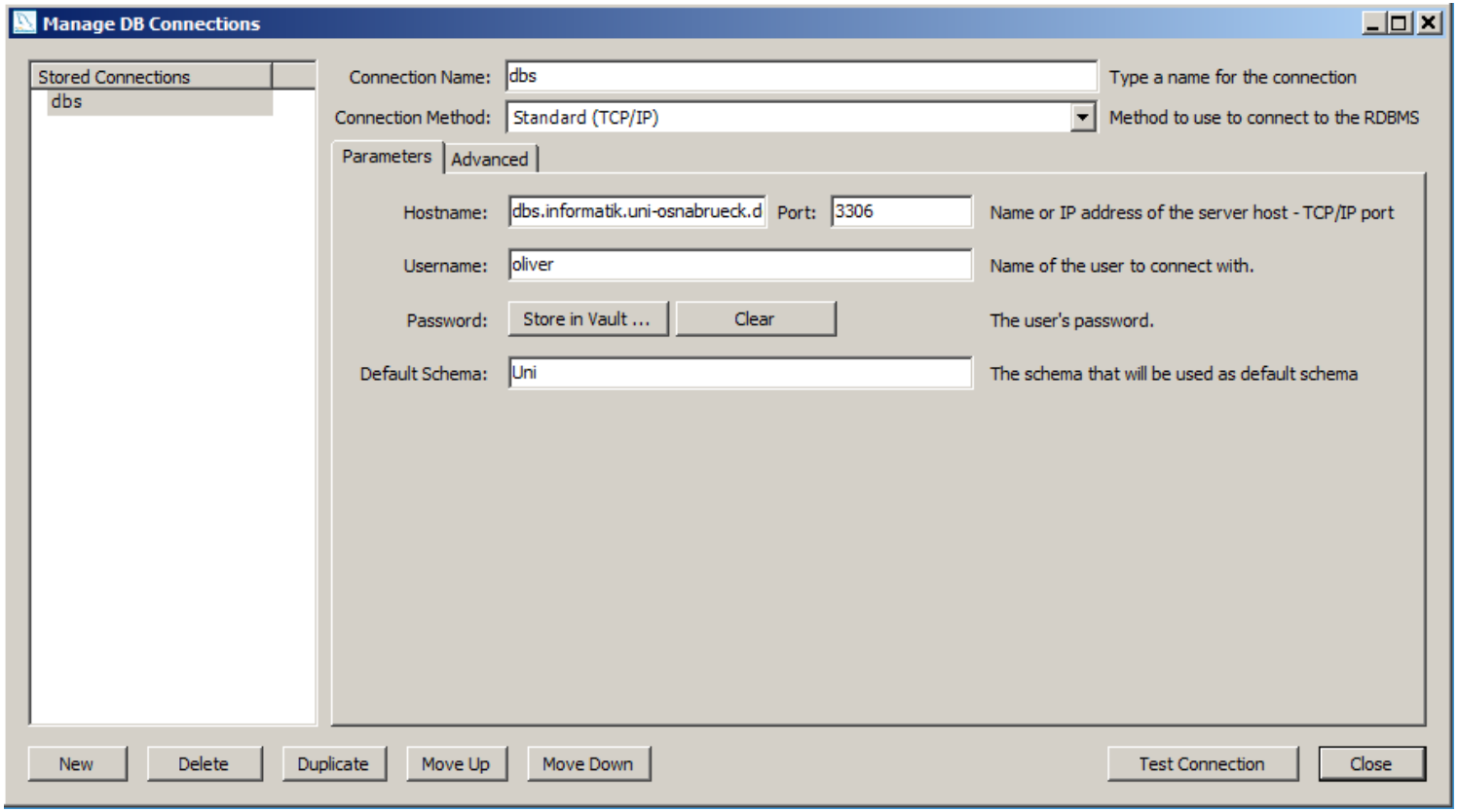
- **L**inux
- **A**pache
- **M**ySQL
- **P**HP

phpmyadmin

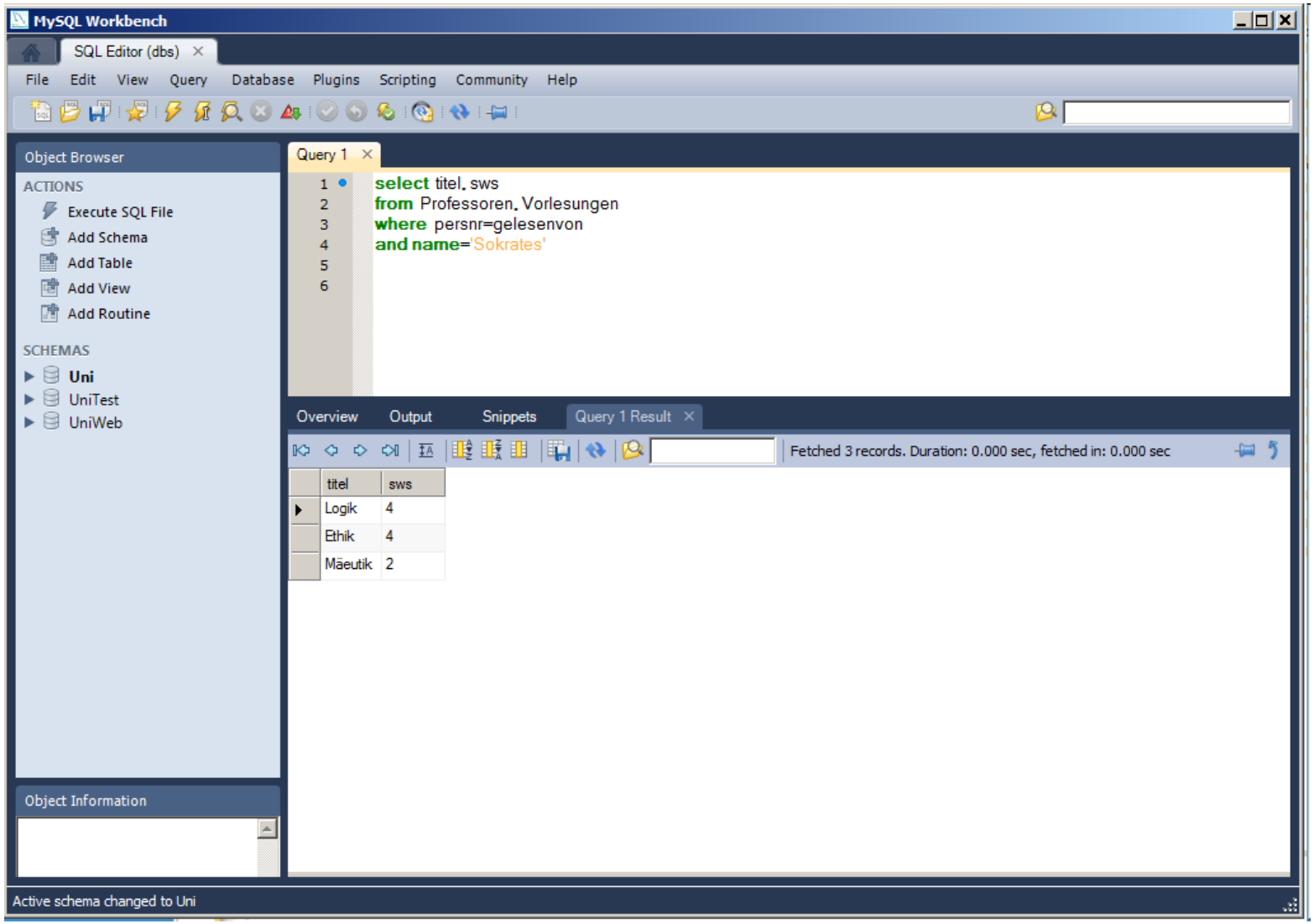


http://dbs.informatik.uni-osnabrueck.de/phpmyadmin

MySQL Workbench: manage connections



MySQL WorkBench: SQL



MySQL Referenzhandbuch

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

SQL: numerische Datentypen

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

Microsoft SQL Server:

8	**money**	Währungswerte mit 4 Nachkommastellen (MS SQL)

SQL Datentypen für Zeitangaben

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

SQL: Datentypen für Zeichenketten

n	**`char(n)`**	Zeichenkette fester Länge mit n ≤ 255 Zeichen
n+d	**`varchar(n)`**	Zeichenkette variabler Länge mit n ≤ 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 ≤ 255 Bytes

n+d **varbinary(n)** Binärdaten variabler Länge
mit n ≤ 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
from
where
order by
asc
desc
as
like
upper
lower

distinct
count
sum
avg
max
min
group by
having

in
not
null
exists
all
some

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

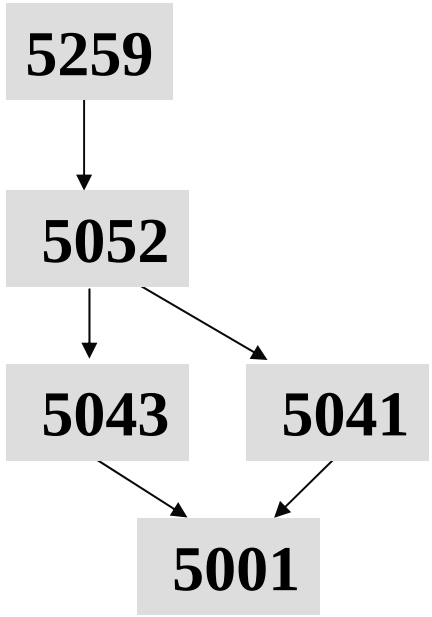
34.) Liste Studenten, die alle 4-stündigen Vorlesungen hören:

```
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" (vorlnr 5259)

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



⇒ Transitive Hülle einer rekursiven Relation

Prolog: Transitive 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'))
```

Oracle: Transitive Hülle

```
select Titel
from   Vorlesungen
where  VorlNr in (
         select Vorgaenger
         from voraussetzen
         connect by Nachfolger = prior Vorgaenger
         start with Nachfolger = (
                  select VorlNr
                  from Vorlesungen
                  where Titel = 'Der Wiener Kreis'
         )
)
```

MySQL: Transitive Hülle

```
create temporary table nach (nr integer)
create temporary table vor (nr integer)

insert into nach
      select vorlnr from Vorlesungen
      where titel = 'Der Wiener Kreis'
```

mehrfach iterieren:

```
insert into vor
    select v.vorgaenger
    from voraussetzen v
    where v.nachfolger in (select * from nach)

delete from nach
insert into nach
    select distinct nr from vor
```

SQL: insert

Workbench sql3

1.) Füge neue Vorlesung mit einigen Angaben ein:

```
insert into Vorlesungen (VorlNr, Titel, gelesenVon)
values (4711, 'Selber Atmen', 2125)
```

2.) Schicke alle Studenten in die Vorlesung *Selber Atmen*:

```
insert  into hoeren
select  MatrNr, VorlNr
from    Studenten, Vorlesungen
where   Titel = 'Selber Atmen'
```

SQL: update

3.) Erweitere die Vorlesung um ihre Semesterwochenstundenzahl:

```
update  Vorlesungen
set     SWS=6
where   Titel='Selber Atmen'
```

Erhöhe die Semesterzahl bei allen Studenten

```
update Studenten
set semester = semester + 1
```

Trage bei allen Professoren die Zahl ihrer Hörer ein:

```
update Professoren p set anzhoerer =
(select count(*)
        from Vorlesungen v, hoeren h
        where p.PersNr = v.gelesenVon
        and h.VorlNr = v.VorlNr)
```

SQL:delete

4.) Entferne alle Studenten aus der Vorlesung *Selber Atmen*:

```
delete from hoeren
where vorlnr=
      (select VorlNr from Vorlesungen
       where Titel = 'Selber Atmen')
```

5.) Entferne die Vorlesung *Selber Atmen*:

```
delete from Vorlesungen
where titel = 'Selber Atmen'
```

SQL: Sichten

1.) Lege Sicht an für Prüfungen ohne Note:

```
create view pruefenSicht as
select MatrNr, VorlNr, PersNr
from   pruefen
```

2.) Lege Sicht an für Studenten und ihre Professoren:

```
create  view StudProf
(Sname, Semester, Titel, Pname) as
select  s.Name, s.Semester, v.Titel, p.Name
from    Studenten s, hoeren h, Vorlesungen v, Professoren p
where   s.MatrNr       = h.MatrNr
and     h.VorlNr       = v.VorlNr
and     v.gelesenVon   = p.PersNr
```

SQL: Sichten

3.) Lege Sicht an für Professoren und Durchschnittsnote:

```
create view ProfNote (Persnr, Durchschnittsnote) as
select PersNr, avg(Note)
from pruefen
group by persnr
```

4.) Entferne Sichten wieder

```
drop view PruefenSicht;
drop view StudProf;
drop view ProfNote;
```

SQL: Generalisierung durch Verbund

5.) Lege Untertyp als Verbund von Obertyp und Erweiterung an:

```
create table Angestellte   (PersNr      integer not null,
                            Name       varchar(30) not null)
create table ProfDaten     (PersNr      integer not null,
                            Rang       character(2),
                            Raum       integer)
create table AssiDaten     (PersNr      integer not null,
                            Fachgebiet varchar(30),
                            Boss       integer)
create view Profs as
   select a.persnr, a.name, d.rang, d.raum
   from   Angestellte a, ProfDaten d
   where  a.PersNr = d.PersNr
create view Assis as
   select a.persnr, a.name, d.fachgebiet, d.boss
   from   Angestellte a, AssiDaten d
   where  a.PersNr = d.PersNr
```

SQL: Tabellen und Sichten entfernen

Entferne die Tabellen und Sichten wieder:

```
drop  table Angestellte
drop  table AssiDaten
drop  table ProfDaten
drop  view  Profs
drop  view  Assis
```

SQL: Generalisierung durch Vereinigung

6.) Lege Obertyp als Vereinigung von Untertypen an (zwei der drei Untertypen sich schon vorhanden):

```
create table AndereAngestellte (PersNr integer not null,
                                Name varchar(30) not null)

create view Angestellte as
   (select PersNr, Name from Professoren) union
   (select PersNr, Name from Assistenten) union
   (select PersNr, Name from AndereAngestellte)
```

Entferne die Tabelle und die Sichten wieder:

```
drop table andereAngestellte
drop view Angestellte
```

Index

```
create index titelindex
on Vorlesungen(titel asc)

create index semesterindex
on Studenten(semester asc)

drop index titelindex on Vorlesungen
drop index semesterindex on Studenten
```

Bulkinsert

4711;Willi;C4;339;1951-03-24
4712;Erika;C3;222;1962-09-18

```
LOAD DATA INFILE '/tmp/prof.txt'
INTO TABLE Professoren
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
```

Tabellen konto, gebucht, abgelehnt

```
create table konto (nr int, stand int)

insert into konto values (1,100)
insert into konto values (2,100)
insert into konto values (3,100)

create table gebucht (
  datum date, nr_1 int, nr_2 int, betrag int)

create table abgelehnt (
  datum date, nr_1 int, nr_2 int, betrag int)
```

Stored Procedure ueberweisung

```
drop procedure if exists ueberweisung
create procedure ueberweisung (x int, y int, betrag int)
begin
  set @s = (select stand from konto where nr = x);
  if (@s < betrag)
  then
    insert into abgelehnt values (now(), x, y, betrag);
  else
    update konto set stand = stand-betrag where nr = x;
    update konto set stand = stand+betrag where nr = y;
    insert into gebucht values (now(), x, y, betrag);
  end if;
end
```

```
call ueberweisung(2,3,50)
```

Stored Function f2c

```
drop function if exists f2c

create function f2c (fahrenheit int)
returns int
deterministic
begin
  set @celsius=(fahrenheit-32)/9.0 * 5.0;
  return @celsius;
end
```

```
select celsius, f2c(celsius) from klimatabelle
```

Stored Function ggt

```
drop function if exists ggt

create function ggt(a int, b int)
returns int
deterministic
begin
  while a != b DO
    if (a>b)
      then set a = a-b;
      else set b = b-a;
    end if;
  end while;
  return a;
end
```

```
select ggt(315,60)
```

Stored Procedure berechneVorlesungen

```
drop procedure if exists berechneVorlesungen
create procedure berechneVorlesungen() -- speichere pro Professor
begin                                  -- die Zahl seiner Vorlesungen
  declare done int default 0;
  declare prof_name CHAR(16);
  declare prof_cursor cursor for
    select p.name
    from Professoren p, Vorlesungen v
    where p.persnr = v.gelesenvon;
  declare continue handler for sqlstate '02000' set done = 1;
  update Professoren set anzVorlesungen = 0;
  open prof_cursor;
  repeat
    fetch prof_cursor into prof_name;
    if not done then
      update Professoren set anzVorlesungen = anzVorlesungen + 1
      where name = prof_name;
    end if;
  until done end repeat;
  close prof_cursor;
end
```

```
call berechneVorlesungen()
```

Stored Procedure zuwenighoerer

```
DROP PROCEDURE if exists zuwenigHoerer

CREATE PROCEDURE zuwenigHoerer() -- setze Professor auf Rang C2
begin                            -- falls weniger als 2 Hoerer
  declare done int default 0;
  declare nr, anzahl int;
  declare prof_cursor cursor for
    select gelesenvon, count(*) from Vorlesungen v, hoeren h
    where v.vorlnr = h.vorlnr
    group by gelesenvon;
  declare continue handler for sqlstate '02000' set done = 1;
  open prof_cursor;
  fetch prof_cursor into nr, anzahl;
  while not done do
    if (anzahl <2) then
      update Professoren set rang = 'C2' where persnr = nr;
    end if;
    fetch prof_cursor into nr, anzahl;
  end while;
  close prof_cursor;
end
```

Stored Procedure für Transitive Hülle

```
create procedure transitivehuelle (start int)
begin
  create temporary table nach (nr integer);
  create temporary table vor (nr integer);
  set @alt = 0; set @neu = 0;
  insert into nach values (start);
  repeat
    insert into vor
      select distinct v.vorgaenger
        from voraussetzen v
        where v.nachfolger in (select * from nach);
    delete from nach;
    insert into nach select distinct nr from vor;
    @alt = @neu;
    @neu = (select count(*) from nach);
    UNTIL (@neu = @alt) END REPEAT;
    select distinct * from vor;
end
```

```
call transitivehuelle(5259);
```

Workbench