Datenbanksysteme SS 2007

Frank Köster
(Oliver Vornberger)

Institut für Informatik
Universität Osnabrück

Kapitel 7a:
Structured Query Language (SQL)

MySQL 5.0

(enthalten in: http://www.apachefriends.org/de/xampp.html)

XAMPP

So mancher wird schon die Erfahrung gemacht haben: Ein Apache-Webserver installiert sich nicht so leicht. Noch schwieriger wird es, wenn weitere Pakete wie MySQL, PHP oder Perl dazukommen.

XAMPP ist eine Distribution von Apache, MySQL, PHP und Perl, die es ermöglicht diese Programme auf sehr einfache Weise zu installieren.

Zur Zeit gibt es vier XAMPP-Distributionen:

Installation (1/8)

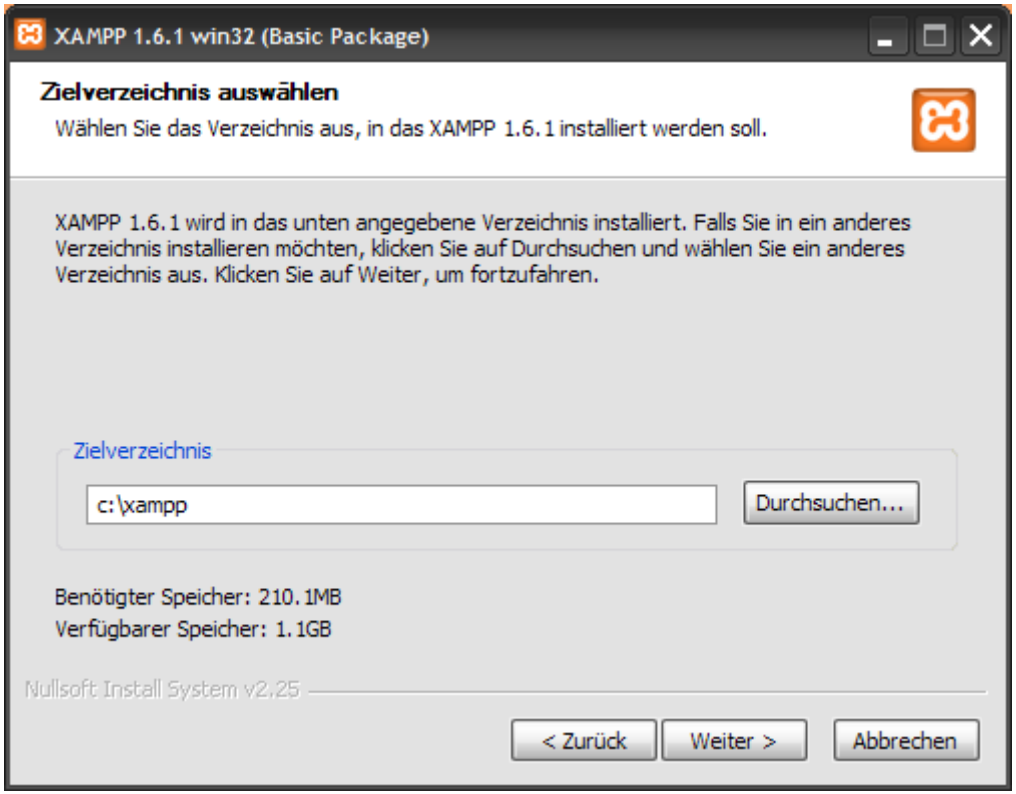
Download des Installers …
(http://www.apachefriends.org/de/xampp.html)

xampp-win32-1.6.1-...

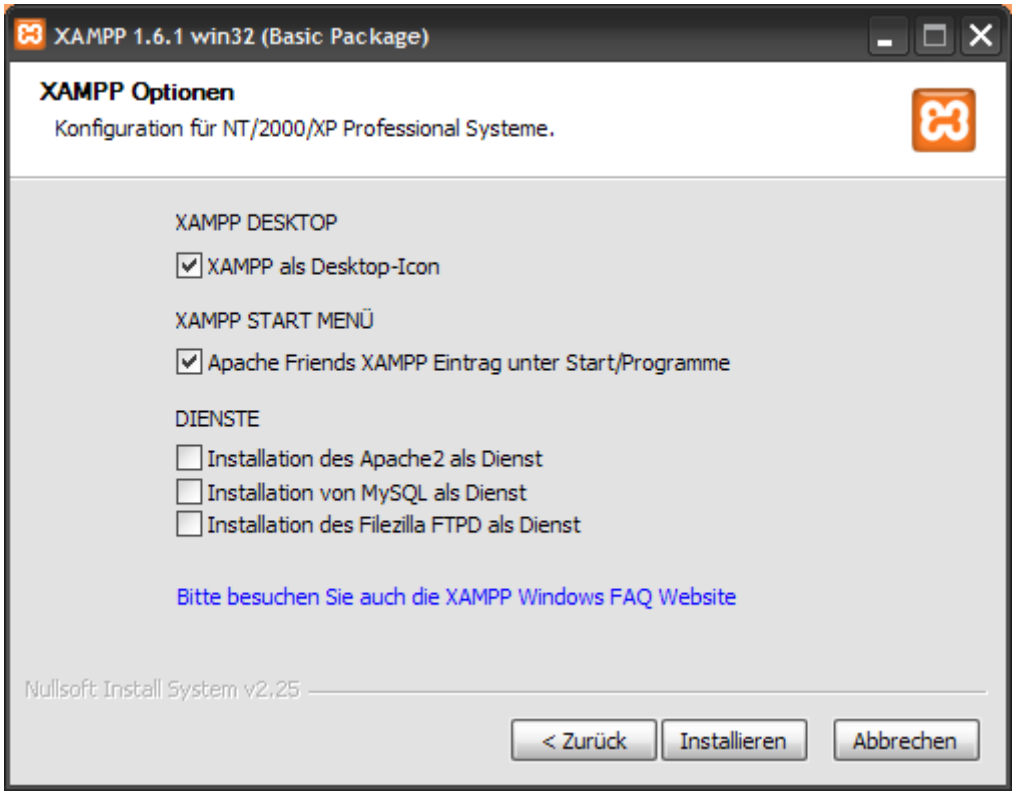
Start des Installers

Installation (2/8)

Installation (3/8)

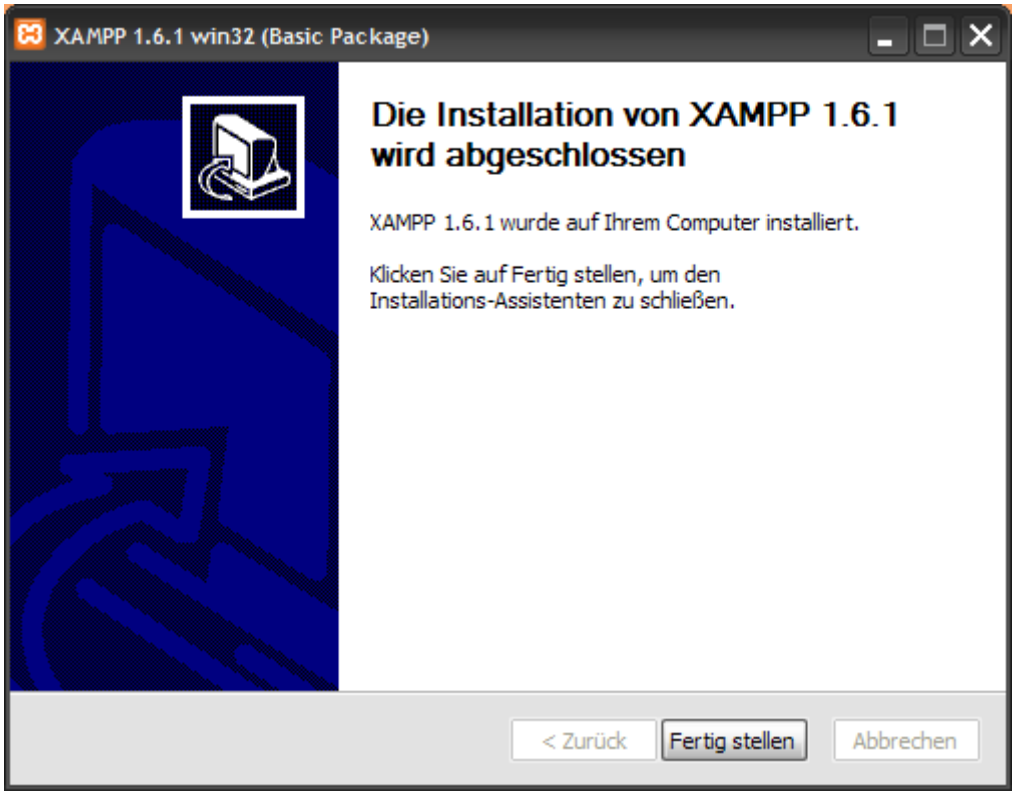


Installation (4/8)

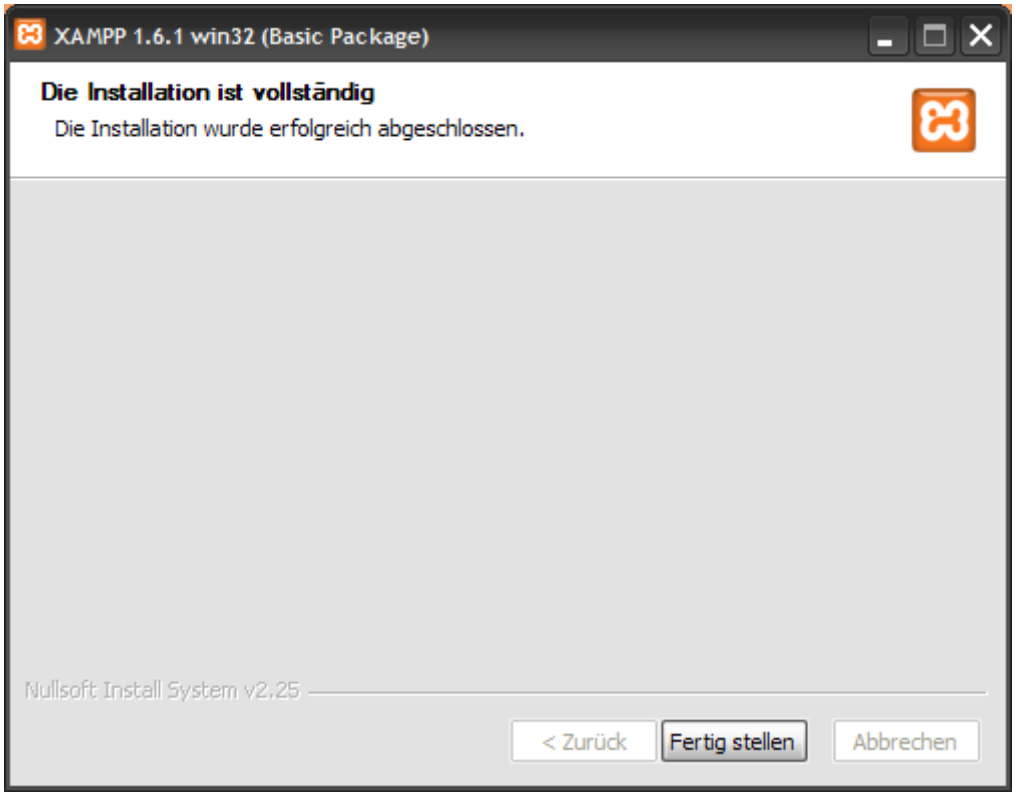


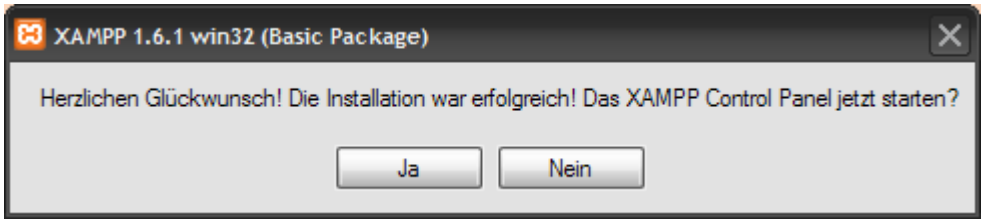
[...]

Installation (5/8)

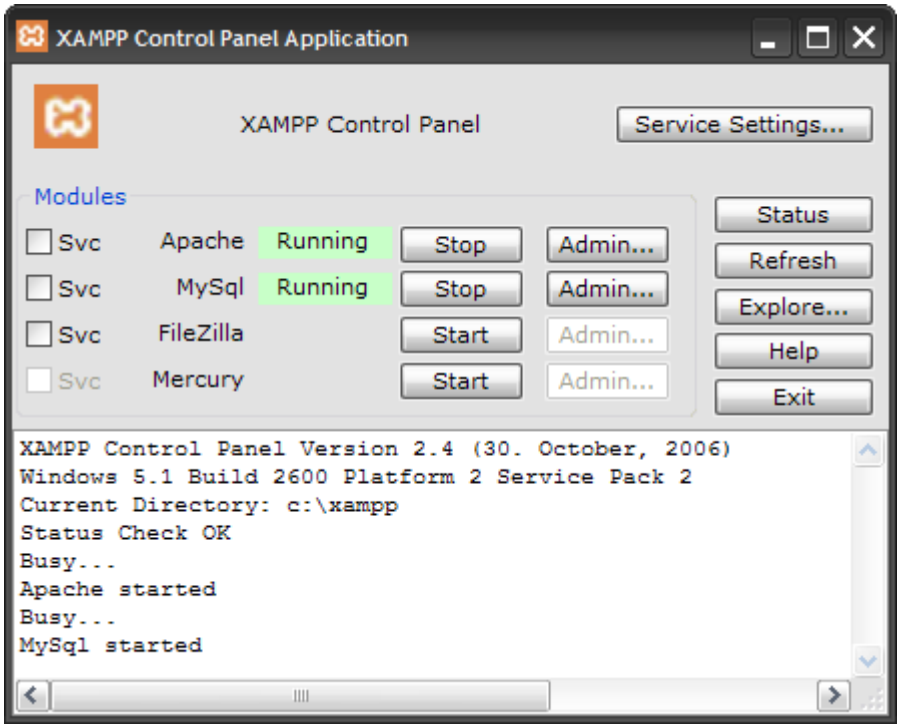


Installation (6/8)

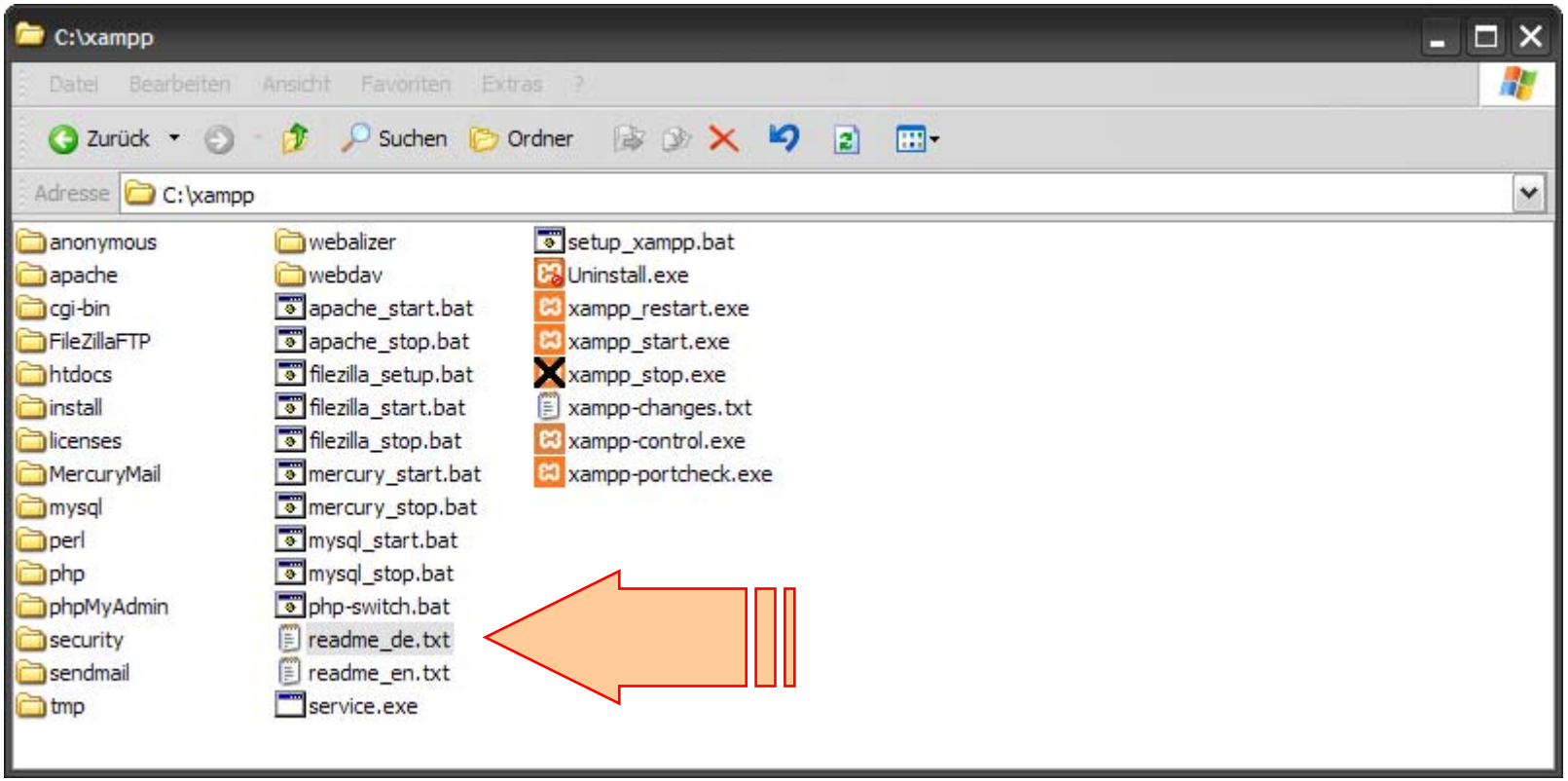




Installation (7/8)



Installation (8/8)



SQL: Geschichte

1970: System R mit Abfragesprache Sequel

1992: SQL-92 = SQL 2

1999: SQL:1999 = SQL 3 (objektorientiert)

SQL: Datentypen (1/3)

8	bigint	ganze Zahlen von -2^{63} bis $+2^{63}$
4	int	ganze Zahlen von -2^{31} bis $+2^{31}$
2	smallint	ganze Zahlen von -2^{15} bis $+2^{15}$
1	tinyint	ganze Zahlen von 0 bis 255
1	bit	ganze Zahlen von 0 bis 1
n	decimal(n,k)	numerische Daten, feste Genauigk. von -10^{38} bis $+10^{38}$ Anzeige mit n Stellen, davon k nach dem Komma
n	numeric(n,k)	entspricht decimal
4	real	Gleitkommazahlen von -10^{38} bis $+10^{38}$
8	float	Gleitkommazahlen von -10^{308} bis $+10^{308}$
8	money	Währungsdatenwerte von -2^{63} bis $+2^{63}$ Anzeige: 4 Nachkommastellen
4	smallmoney	Währungsdatenwerte von -2^{15} bis $+2^{15}$

SQL: Datentypen (2/3)

char(n)	String fester Länge mit maximal 8.000 Zeichen
varchar(n)	String variabler Länge mit maximal 8.000 Zeichen
nchar(n)	Unicode-Daten fester Länge mit maximal 4.000 Zeichen
nvarchar(n)	Unicode-Daten variabler Länge mit maximal 4.000 Zeichen
datetime	Zeitangaben der Form TT.MM.YYYY(SS:MM:…)

SQL: Datentypen (3/3)

blob	Binärdaten variabler Länge
image	Bilddaten variabler Länge
audio	Audiodaten variabler Länge
video	Videodaten variabler Länge
text	Textdaten variabler Länge
xmltype	XML-Darstellungen variabler Länge

SQL: create (1/3)

```
CREATE TABLE Studenten
       (MatrNr      INTEGER PRIMARY KEY,
        Name        VARCHAR(20) NOT NULL,
        Semester    INTEGER,
        GebDatum    DATETIME);

CREATE TABLE Professoren
       (PersNr      INTEGER PRIMARY KEY,
        Name        VARCHAR(20) NOT NULL,
        Rang        CHAR(2) CHECK (Rang in ('C2','C3','C4')),
        Raum        INTEGER UNIQUE,
        Gebdatum    DATETIME);
```

SQL: create (2/3)

```
CREATE TABLE Assistenten
       (PersNr       INTEGER PRIMARY KEY,
        Name         VARCHAR(20) NOT NULL,
        Fachgebiet   VARCHAR(20),
        Boss         INTEGER REFERENCES Professoren,
        GebDatum     DATETIME);

CREATE TABLE Vorlesungen
       (VorlNr       INTEGER PRIMARY KEY,
        Titel        VARCHAR(20),
        SWS          INTEGER,
        gelesenVon   INTEGER REFERENCES Professoren);
```

SQL: create (3/3)

```
CREATE TABLE hoeren
       (MatrNr         INTEGER REFERENCES Studenten   ON UPDATE CASCADE
                                                      ON DELETE CASCADE,
        VorlNr         INTEGER REFERENCES Vorlesungen ON UPDATE CASCADE,
        PRIMARY KEY    (MatrNr, VorlNr));

CREATE TABLE voraussetzen
       (Vorgaenger     INTEGER REFERENCES Vorlesungen  ON UPDATE CASCADE,
        Nachfolger     INTEGER REFERENCES Vorlesungen,
        PRIMARY KEY    (Vorgaenger, Nachfolger));

CREATE TABLE pruefen
       (MatrNr         INTEGER REFERENCES Studenten   ON UPDATE CASCADE
                                                      ON DELETE CASCADE,
        VorlNr         INTEGER REFERENCES Vorlesungen ON UPDATE CASCADE,
        PersNr         INTEGER REFERENCES Professoren,
        Note           NUMERIC(3,1) CHECK (Note between 0.7 and 5.0),
        PRIMARY KEY    (MatrNr, VorlNr));
```

SQL: insert

```
INSERT INTO Studenten(MatrNr, Name, Semester, GebDatum)
VALUES (24002, 'Xenokrates', 18, '1975-10-23');
…

INSERT INTO Professoren(PersNr, Name, Rang, Raum, GebDatum)
VALUES (2125, 'Sokrates', 'C4', 226,'1923-08-23');
…

INSERT INTO Assistenten(PersNr, Name, Fachgebiet, Boss, GebDatum)
VALUES (3002, 'Platon', 'Ideenlehre', 2125, '1966-08-14');
…

INSERT INTO Vorlesungen(VorlNr, Titel, SWS, gelesenVon)
VALUES (5001, 'Grundzüge', 4, 2137);
…

INSERT INTO hoeren(MatrNr, VorlNr)
VALUES (26120, 5001);
…

INSERT INTO voraussetzen(Vorgaenger, Nachfolger)
VALUES (5001, 5041);
…

INSERT INTO pruefen(MatrNr, VorlNr, PersNr, Note)
VALUES (28106, 5001, 2126, 1.0);
…
```

SQL: alter, modify, drop

Tabelle um eine Spalte erweitern:

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

Tabellenspalte ändern:

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

Tabelle um eine Spalte verkürzen:

```
alter table Studenten
      drop column Vorname
```

Tabelle entfernen:

```
drop table Studenten
```

SQL: Schlüsselworte (Auszug)

select
from
where
order by
asc
desc
as
like
upper
lower

distinct
count
sum
avg
max
min
group by
having

is
not
null
exists
all
some

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 aller 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, upper, order, distinct

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

```
select * from Vorlesungen
where upper(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: Verbund

9.) Liste den Dozenten der Vorlesung Logik:

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

10.) 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
where   s.MatrNr = h.MatrNr
and     h.VorlNr = v.VorlNr
```

SQL: Self Join

11.) 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'
```

SQL: avg, group

12.) Liste die durchschnittliche Semesterzahl:

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

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

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

14.) Liste Summe der SWS pro Professor:

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

SQL: having

15.) 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

16.) 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

17.) Liste alle Prüfungen, die als Ergebnis die schlechteste Note haben:

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

18.) Liste alle Professoren zusammen mit ihrer Lehrbelastung:

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

SQL: exists

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

```
select s.*
from   Studenten s
where  exists (select p.*
               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 <

20.) Liste alle Assistenten, die für einen jüngeren Professor arbeiten:

```
select a.*
from   Assistenten a, Professoren p
where  a.Boss = p.PersNr
and    a.GebDatum < p.GebDatum
```

SQL: geschachtelt

21.) 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, s.Name) tmp
where tmp.VorlAnzahl > 2
```

SQL: geschachtelt

22.) Liste die Namen und Geburtstage der Gehaltsklassen-ältesten:

```
select p.Rang, p.Name, p.GebDatum
from   Professoren p,
       (select Rang, min(GebDatum) as maximum
        from Professoren
        group by Rang) tmp
where  p.Rang = tmp.Rang
       and p.GebDatum = tmp.maximum
```

SQL: union

23.) Liste die Vereinigung von Professoren- und Assistenten-Namen:

```
(select Name from Assistenten)
union
(select Name from Professoren)
```

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

```
(select Name from Assistenten)
minus
(select Name from Professoren)
```

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

```
(select Name from Assistenten)
intersect
(select Name from Professoren)
```

SQL: not in / exists

24.) 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

25.) Liste Studenten mit größter Semesterzahl:

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

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

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

SQL: not, exists

27.) 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
             )
        )
```

SQL: transitive Hülle

--.) Transitive Hülle einer rekursiven Relation (nur in Oracle): Liste alle Voraussetzungen für Vorlesung „Der Wiener Kreis“:

```
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'
        )
)
```

Datenbanksysteme SS 2007

Ende von Kapitel 7a: SQL