Datendefinitions-, Manipulations- und Anfrage-Sprache SQL, Datendefinition, Veränderung am Datenbestand, Einfache SQL Abfrage, Anfragen über mehrere Relationen, Mengenfunktionen, Aggregatfunktion und Gruppierung, Geschachtelte Anfragen

DATENBANKSYSTEME: SQL

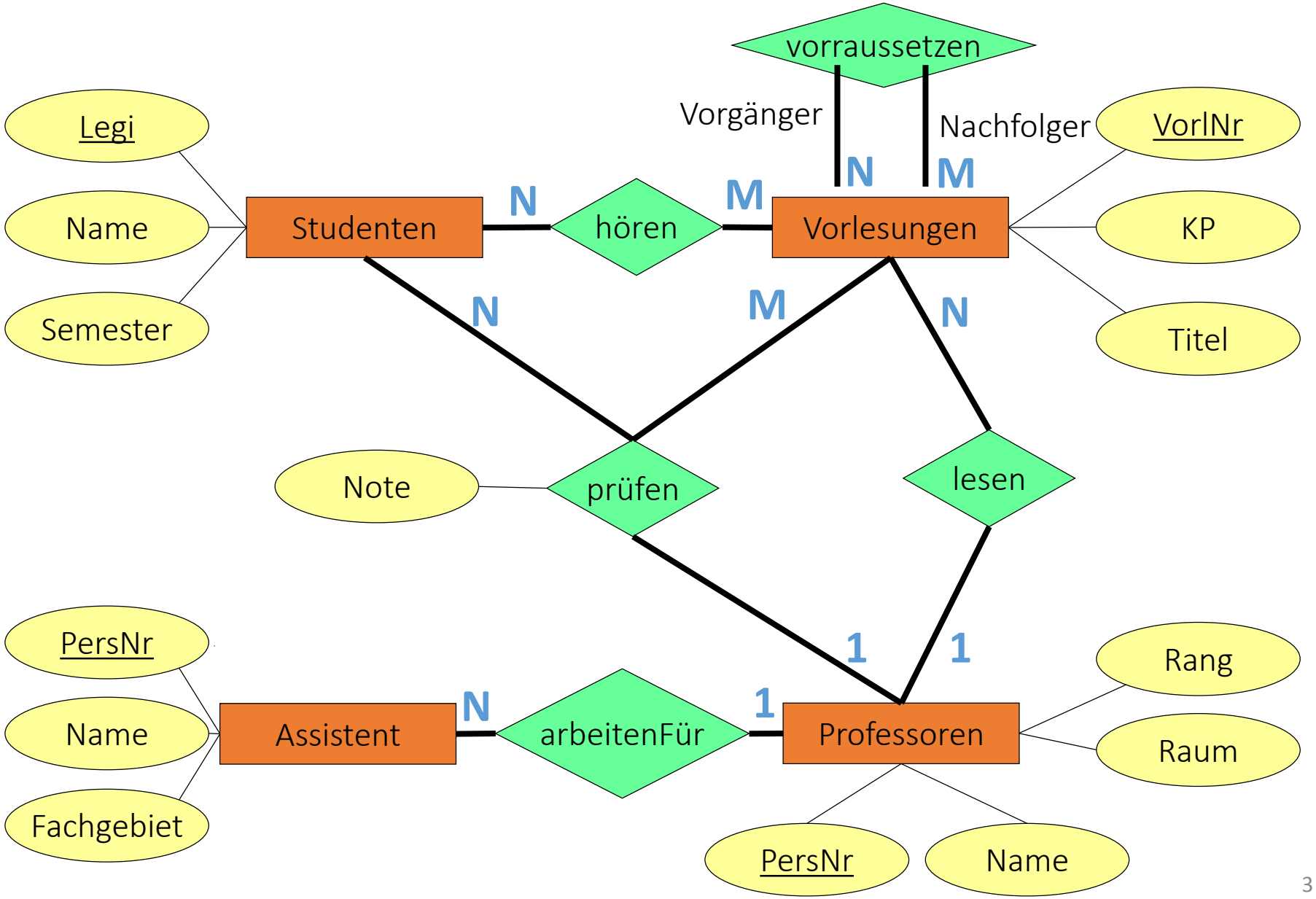
SQL (Structured Query Language)

Nachfolger von
Sequel = Structured English Query Language

Familie von Standards

- Anfrage- (Query)-Sprache – Anfragen
- Datendefinitionssprache (DDL) – Schemas
- Datenmanipulationssprache (DML) – Updates

Wdh. Uni Schema



Relationales Modell der Uni-DB

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	FP	226
2126	Russel	FP	232
2127	Kopernikus	AP	310
2133	Popper	AP	52
2134	Augustinus	AP	309
2136	Curie	FP	36
2137	Kant	FP	7

Studenten		
Legi	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	KP	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
	Die 3 Kritiken	4	2137

hören	
Legi	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022

Assistenten			
PerslNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

prüfen			
Legi	Nr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

SQL: Anzeigen einer existierenden Tabelle

SELECT * FROM Tabellenname

gibt den Inhalt eine Tabelle (Relation) aus

```
SELECT * FROM Professoren
```

PersNr	Name	Rang	Raum
2125	Sokrates	FP	226
2126	Russel	FP	232
2127	Kopernikus	AP	310
2133	Popper	AP	52
2134	Augustinus	AP	309
2136	Curie	FP	36
2137	Kant	FP	7

Einfaches SQL Statement

SELECT *Spaltennamen* FROM *Tabellennamen*

entspricht der Projektion π ohne Duplikatelimination:

```
SELECT Rang FROM Professoren
```

PersNr	Name	Rang	Raum
2125	Sokrates	FP	226
2126	Russel	FP	232
2127	Kopernikus	AP	310
2133	Popper	AP	52
2134	Augustinus	AP	309
2136	Curie	FP	36
2137	Kant	FP	7

Rang
FP
FP
AP
AP
AP
FP
FP

Projektion

SELECT DISTINCT *Spaltennamen* FROM *Tabellennamen*

entspricht der Projektion π:

```
SELECT DISTINCT Rang FROM Professoren
```

PersNr	Name	Rang	Raum
2125	Sokrates	FP	226
2126	Russel	FP	232
2127	Kopernikus	AP	310
2133	Popper	AP	52
2134	Augustinus	AP	309
2136	Curie	FP	36
2137	Kant	FP	7

Rang
FP
AP

$\pi_{\mathbf{Rang}}(\mathbf{Professoren})$

Projektion

Anderes Beispiel

```
SELECT Name, Population, SurfaceArea FROM country
```

Name	Population	SurfaceArea
Aruba	103000	193.00
Afghanistan	22720000	652090.00
Angola	12878000	1246700.00
Anguilla	8000	96.00
Albania	3401200	28748.00
Andorra	78000	468.00
Netherlands Antilles	217000	800.00

$$\pi_{\text{Name,Population,SurfaceArea}}(\text{country})$$

Selektion

SELECT *Spalten* FROM *Tabellen* WHERE *Selektionsprädikat*
entspricht der Selektion σ

```
SELECT Name, Population FROM country WHERE SurfaceArea < 10
```

Name	Population
Gibraltar	25000
Monaco	34000
Holy See (Vatican City State)	1000

$$\pi_{\mathbf{Name,Population,SurfaceArea}}(\sigma_{\mathrm{SurfaceArea}<10}(\mathbf{country}))$$

Sortieren

SELECT *Spalten* FROM *Tabellen* ORDER BY Name[ASC|DESC]

```
SELECT Name, Population, SurfaceArea FROM country ORDER BY
SurfaceArea DESC
```

Name	Population	SurfaceArea
Russian Federation	146934000	17075400.00
Antarctica	0	13120000.00
Canada	31147000	9970610.00
China	1277558000	9572900.00
United States	278357000	9363520.00
Brazil	170115000	8547403.00
Australia	18886000	7741220.00
India	1013662000	3287263.00
Argentina	37032000	2780400.00

$\pi_{\text{Name,Population,SurfaceArea}}(\text{country})$

Kombination

```
SELECT Code, Name, Population, SurfaceArea
FROM country WHERE
Population > 7000000 AND Population < 8000000
ORDER BY SurfaceArea ASC
```

Code	Name	Population	SurfaceArea
RWA	Rwanda	7733000	26338.00
CHE	Switzerland	7160400	41284.00
AZE	Azerbaijan	7734000	86600.00
GIN	Guinea	7430000	245857.00
TCD	Chad	7651000	1284000.00

$$\pi_{\text{Code,Name,Population,SurfaceArea}}(\sigma_{Population>7000000 \wedge Population<8000000}(\text{country}))$$

Umbenennung von Attributen

SELECT ... *Name as NeuerName ...* FROM *Tabellen ...*

entspricht der Umbenennung ρ.

SELECT Titel, **gelesenVon as Dozent** FROM Vorlesungen

VorlNr	Titel	KP	gelesenVon
5001	Grundzuege	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Maeeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126

Titel	Dozent
Grundzuege	2137
Ethik	2125
Erkenntnistheorie	2126
Maeeutik	2125
Logik	2125
Wissenschaftstheorie	2126
Bioethik	2126

$$\pi_{\text{Titel,Dozent}}(\rho_{\text{gelesenVon} \leftarrow \text{Dozent}}(\text{Vorlesungen}))$$

Umbenennung von Tabellen

SELECT ... FROM *Tabellenname Alias* ...

entspricht der Umbenennung ρ.

wozu?
→ kommt gleich!

```
SELECT v.Titel FROM Vorlesungen v
```

VorlNr	Titel	KP	gelesenVon
5001	Grundzuege	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Maeeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126

Titel
Grundzuege
Ethik
Erkenntnistheorie
Maeeutik
Logik
Wissenschaftstheorie
Bioethik

$\pi_{\text{Titel}}(\rho_{\text{V}}(\text{Vorlesungen}))$

Kartesisches Produkt

SELECT ... FROM *Tabelle1, Tabelle2 ...*

entspricht dem kartesischen Produkt

```
SELECT * FROM studenten, hören
```

Legi	Name	Semester	Legi	VorlNr
24002	Xenokrates	18	26120	5001
25403	Jonas	12	26120	5001
26120	Fichte	10	26120	5001
26830	Aristoxenos	8	26120	5001
27550	Schopenhauer	6	26120	5001
28106	Carnap	3	26120	5001
29120	Theophrastos	2	26120	5001
29555	Feuerbach	2	26120	5001
4711	Unbekannter	*NULL*	26120	5001
24002	Xenokrates	18	27550	5001
25403	Jonas	12	27550	5001
26120	Fichte	10	27550	5001

studenten × hören

Kartesisches Produkt plus Selektion

```
SELECT * FROM studenten, hören
WHERE studenten.Legi = hören.Legi
```

Legi	Name	Semester	Legi	VorlNr
25403	Jonas	12	25403	5022
26120	Fichte	10	26120	5001
27550	Schopenhauer	6	27550	4052
27550	Schopenhauer	6	27550	5001
28106	Carnap	3	28106	5041
28106	Carnap	3	28106	5052
28106	Carnap	3	28106	5216
28106	Carnap	3	28106	5259
29120	Theophrastos	2	29120	5001
29120	Theophrastos	2	29120	5041
29120	Theophrastos	2	29120	5049
29555	Feuerbach	2	29555	5001
29555	Feuerbach	2	29555	5022

$$\sigma_{\text{studenten.legi} = \text{hören.Legi}}(\text{studenten} \times \text{hören})$$

Dafür die Umbenennung:

```
SELECT * FROM studenten s, hören h
WHERE s.Legi = h.Legi
```

Legi	Name	Semester	Legi	VorlNr
25403	Jonas	12	25403	5022
26120	Fichte	10	26120	5001
27550	Schopenhauer	6	27550	4052
27550	Schopenhauer	6	27550	5001
28106	Carnap	3	28106	5041
28106	Carnap	3	28106	5052
28106	Carnap	3	28106	5216
28106	Carnap	3	28106	5259
29120	Theophrastos	2	29120	5001
29120	Theophrastos	2	29120	5041
29120	Theophrastos	2	29120	5049
29555	Feuerbach	2	29555	5001
29555	Feuerbach	2	29555	5022

$$\sigma_{\text{s.Legi} = \text{h.Legi}}(\rho_s(\text{studenten}) \times \rho_h(\text{hören}))$$

Join

SELECT ... FROM *Tabelle1 NATURAL JOIN Tabelle2* ...

entspricht dem Join ⋈

```
SELECT *
FROM studenten NATURAL JOIN hören
```

Legi	Name	Semester	VorlNr
25403	Jonas	12	5022
26120	Fichte	10	5001
27550	Schopenhauer	6	4052
27550	Schopenhauer	6	5001
28106	Carnap	3	5041
28106	Carnap	3	5052
28106	Carnap	3	5216
28106	Carnap	3	5259
29120	Theophrastos	2	5001
29120	Theophrastos	2	5041
29120	Theophrastos	2	5049
29555	Feuerbach	2	5001
29555	Feuerbach	2	5022

studenten ⋈ hören

Unterschied zur vorigen Folie:
Legi kommt nur einmal vor in der Tabelle.
Join fasst die gleichnamigen Attribute zusammen.

Anfragen über mehrere Relationen

Welcher Professor liest "Mäeutik"?

```
select    Name,  Titel
from      Professoren , Vorlesungen
where     PersNr  =  gelesenVon and  Titel = "Mäeutik" ;
```

Projektion

Kreuzprodukt

Selektion

$$\Pi_{\text{Name,Titel}}(\sigma_{\text{PersNr}=\text{gelesenVon} \wedge \text{Titel}='\text{Mäeutik}'}(\text{Professoren} \times \text{Vorlesungen}))$$

Anfragen über mehrere Relationen

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
⋮	⋮	⋮	⋮
2137	Kant	C4	7

Vorlesungen			
VorlNr	Titel	KP	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
⋮	⋮	⋮	⋮
5049	Mäeutik	2	2125
⋮	⋮	⋮	⋮
4630	Die 3 Kritiken	4	2137

Verknüpfung X

Anfragen über mehrere Relationen (Fortsetzung)

PersNr	Name	Rang	Raum	VorlNr	Titel	KP	gelesen Von
2125	Sokrates	C4	226	5001	Grundzüge	4	2137
1225	Sokrates	C4	226	5041	Ethik	4	2125
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2125	Sokrates	C4	226	5049	Mäeutik	2	2125
⋮	⋮		⋮	⋮	⋮	⋮	⋮
2126	Russel	C4	232	5001	Grundzüge	4	2137
2126	Russel	C4	232	5041	Ethik	4	2125
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2137	Kant	C4	7	4630	Die 3 Kritiken	4	2137

↓ **Auswahl σ**

PersNr	Name	Rang	Raum	VorlNr	Titel	KP	gelesen Von
2125	Sokrates	C4	226	5049	Mäeutik	2	2125

↓ **Projektion π**

Name	Titel
Sokrates	Mäeutik

String-Vergleiche mit like

Platzhalter "%" ; "_"
"%" steht für beliebig viele (auch gar kein) Zeichen
"_" steht für genau ein Zeichen

```
select *
from Studenten
where Name like `T%eophrastos‘;
```

```
select distinct s.Name
from Vorlesungen v, hören h, Studenten s
where s.MatrNr = h.MatrNr and h.VorlNr = v.VorlNr  and  v.Titel like `%thik%‘;
```

Syntactic Sugar

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

select *
from Studenten
where Semester between 1 and 4;

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

Anfragen: weitere Beispiele

Welche Studenten hören welche Vorlesungen?

```
select   Name, Titel
from     Studenten, hören, Vorlesungen
where    Studenten.Legi = hören.Legi
         and hören.VorlNr = Vorlesungen.VorlNr
```

Alternative:

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

Anfragen: weitere Beispiele

Welche Studenten kennen sich aus Vorlesungen?

```
SELECT s1.name, s2.name
FROM   Studenten s1, Studenten s2, hören h1, hören h2
WHERE  s1.Legi = h1.Legi
       AND s2.Legi = h2.Legi
       AND h1.VorlNr = h2.VorlNr
       AND s1.Name < s2.Name
```

Anfragen: weitere Beispiele

In welchen Städten wird Deutsch verstanden?

```
SELECT s.name, c.name
FROM   city s, country c, countrylanguage l
WHERE  s.countrycode = c.code
       AND c.code = l.countrycode
       AND l.language = "German"
       AND l.isOfficial = TRUE
```

Anfragen: weitere Beispiele

In welchen Ländern gibt es Städte mittlerer Grösse (100-500k Einwohner), deren Namen mit "Z" beginnen ?

```
SELECT c.name
FROM   city s, country c
WHERE  s.countrycode = c.code
       AND s.name like "Z%"
       AND s.population between 100000 and 500000
```

Mengenoperationen

Mengenoperation **union**

(**select** PersNr, Name **from** Assistenten)
union
(**select** PersNr, Name **from** Professoren);

union funktioniert nur auf Relationen mit gleichem Schema!

Mengenvergleich

```
select p.Name
from Professoren p
where p.PersNr not in ( select v.gelesenVon
                        from Vorlesungen v);
```

transient erzeugte, separate Tabelle. Klammerung erlaubt Bezug.

Aggregatfunktion und Gruppierung

Aggregatfunktionen **avg, max, min, count, sum**

```
select avg(Semester)
from Studenten;

select v.gelesenVon, sum(v.KP)
from Vorlesungen v
group by v.gelesenVon;
```

Aggregatfunktion und Gruppierung

```
select v.gelesenVon, p.Name, sum(v.KP)
from Vorlesungen v, Professoren p
where v.gelesenVon = p.PersNr and p.Rang = 'FP'
group by v.gelesenVon, p.Name
having avg(v.KP) >= 3;
```

SQL weiss nicht, dass sich der Name innerhalb der Gruppe nicht ändern kann, wenn nur gelesenVon angegeben ist

Besonderheiten bei Aggregatoperationen

SQL erzeugt pro Gruppe ein Ergebnistupel

Deshalb müssen alle in der **select**-Klausel aufgeführten Attribute - außer den aggregierten – auch in der **group by**-Klausel aufgeführt werden

Nur so kann SQL sicherstellen, dass sich das Attribut nicht innerhalb der Gruppe ändert

Abarbeitung der Anfrage in SQL

1. Schritt: *Kreuzprodukt und Selektion*
   from Vorlesungen, Professoren
   where gelesenVon = PersNr **and** Rang = ´FP´
2. Schritt: *Gruppierung*
   group by gelesenVon, Name
3. Schritt: *Selektion der Gruppierung*
   having avg (KP) >= 3
4. Schritt: *Projektion*
   select gelesenVon, Name, **sum** (KP)

Ausführen einer Anfrage mit group by

Vorlesung x Professoren							
VorlNr	Titel	KP	gelesen Von	PersNr	Name	Rang	Raum
5001	Grundzüge	4	2137	2125	Sokrates	FP	226
5041	Ethik	4	2125	2125	Sokrates	FP	226
…	…	…	…	…	…	…	…
4630	Die 3 Kritiken	4	2137	2137	Kant	FP	7

↓ **where**-Bedingung

nach Selektion

VorlNr	Titel	KP	gelesen Von	PersNr	Name	Rang	Raum
5001	Grundzüge	4	2137	2137	Kant	C4	7
5041	Ethik	4	2125	2125	Sokrates	C4	226
5043	Erkenntnistheorie	3	2126	2126	Russel	C4	232
5049	Mäeutik	2	2125	2125	Sokrates	C4	226
4052	Logik	4	2125	2125	Sokrates	C4	226
5052	Wissenschaftstheorie	3	2126	2126	Russel	C4	232
5216	Bioethik	2	2126	2126	Russel	C4	232
4630	Die 3 Kritiken	4	2137	2137	Kant	C4	7

↓ **Gruppierung**

nach Gruppierung

VorlNr	Titel	KP	gelesenVon	PersNr	Name	Rang	Raum
5041	Ethik	4	2125	2125	Sokrates	FP	226
5049	Mäeutik	2	2125	2125	Sokrates	FP	226
4052	Logik	4	2125	2125	Sokrates	FP	226
5043	Erkenntnistheorie	3	2126	2126	Russel	FP	232
5052	Wissenschaftstheo.	3	2126	2126	Russel	FP	232
5216	Bioethik	2	2126	2126	Russel	FP	232
5001	Grundzüge	4	2137	2137	Kant	FP	7
4630	Die 3 Kritiken	4	2137	2137	Kant	FP	7

↓ **having-Bedingung**

VorlNr	Titel	KP	gelesenVon	PersNr	Name	Rang	Raum
5041	Ethik	4	2125	2125	Sokrates	FP	226
5049	Mäeutik	2	2125	2125	Sokrates	FP	226
4052	Logik	4	2125	2125	Sokrates	FP	226
5001	Grundzüge	4	2137	2137	Kant	FP	7
4630	Die 3 Kritiken	4	2137	2137	Kant	FP	7

↓ **Aggregation (sum) und Projektion**

Ergebnis

gelesenVon	Name	**sum** (KP)
2125	Sokrates	10
2137	Kant	8

Geschachtelte Anfrage

Unteranfrage in der **where**-Klausel

Welche Prüfungen sind besser als durchschnittlich verlaufen?

```
select *
   from prüfen
   where Note < ( select avg (Note)
                  from prüfen );
```

Andere Beispiele

Erdbevölkerung

```
select  sum(c.population)
from    country c
```

Andere Beispiele

Bevölkerung der Kontinente

```
select  c.continent, sum(c.population)
from    country c
group by c.continent
```

Andere Beispiele

Tabelle der durchschnittlichen Anzahl Bewohner der Städte eines Landes

```
select  c.name, avg(s.population) as "Stadtdurchschnitt"
from    city s, country c
where   c.code = s.countrycode
group by c.name
```

Andere Beispiele

Tabelle aller Länder mit mehr als 10 Städten mit 1 Mio Einwohnern.
Tabelle soll auch die Anzahl Städte aufzählen.

```
select  c.name, count(c.name)
from    country c, city s
where   s.countrycode = c.code
        and s.population > 1000000
group   by c.name
having  count(c.name)>10
```

Andere Beispiele

Namen aller Länder, in denen als offizielle Sprache **nur** Deutsch gesprochen wird

```
select c.name from
(
select l.countrycode, l.language, count(l.language)
from countrylanguage l
where l.isofficial = TRUE
group by l.countrycode
having count(l.language)=1
) as p, country c
where p.language = "German" and c.code = p.countrycode
```

Weitere Ideen?

Datendefinition (DDL) in SQL

Datentypen

character *(n)*, ***char*** *(n)*

character varying *(n)*, **varchar** *(n)*

numeric *(p,s)*, **integer, decimal**

blob oder **raw** für sehr große binäre Daten

clob für sehr große String-Attribute

date für Datumsangaben

xml für XML-Dokumente

DDL (Fortsetzung)

Anlegen einer Tabelle

```
create table Professoren
                    (PersNr          integer not null,
                     Name  varchar (10) not null
                     Rang    character (2) );
```

Tabelle löschen

```
drop table Professoren;
```

Tabellenstruktur anpassen

```
alter table Professoren add column Raum integer;
alter table Professoren modify column Name varchar(30);
```

Veränderung am Datenbestand

Einfügen von Tupeln

insert into Studenten (Legi, Name)

values (28121, `Archimedes‘);

Studenten		
MatrNr	Name	Semester
29120	Theophrastos	2
29555	Feuerbach	2
28121	Archimedes	-

Null-Wert

insert into hören

select MatrNr, VorlNr

from Studenten, Vorlesungen

where Titel= `Logik‘ ;

Veränderungen am Datenbestand

Löschen von Tupeln

```
delete Studenten
        where Semester > 100;
```

Verändern von Tupeln

```
update Studenten
        set Semester= Semester + 1;
```