

Relationale Anfragesprachen

- Structured Query Language: SQL
- Query by Example: QBE



Kapitel 4

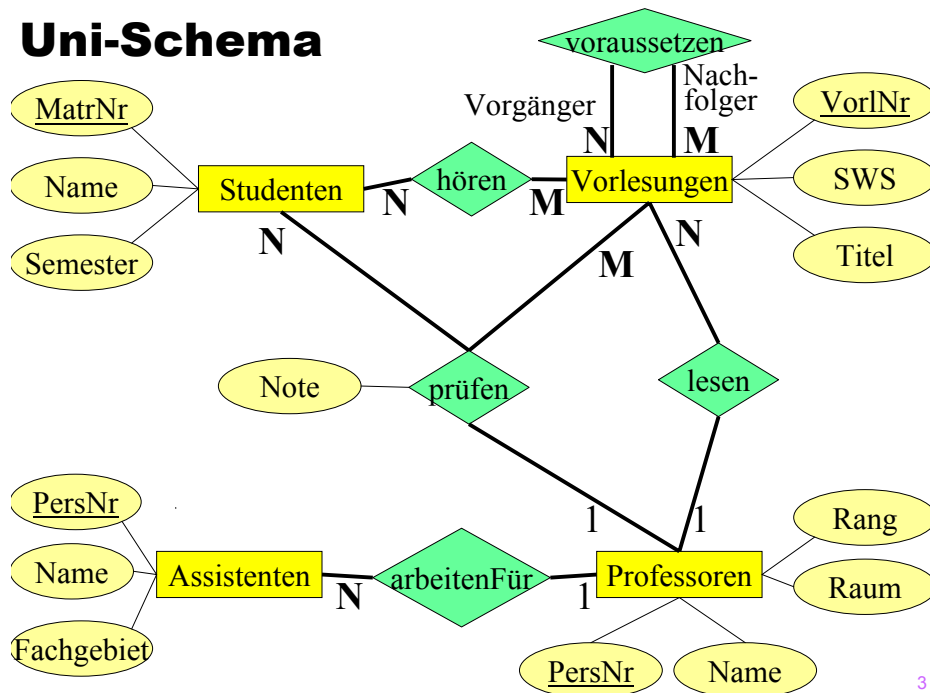
1

SQL

- standardisierte
 - Datendefinitions (DDL)-
 - Datenmanipulations (DML)-
 - Anfrage (Query)-Sprache
- derzeit aktueller Standard ist SQL 99
 - objektrelationale Erweiterung
- Die Syntax ist in Kap. 4 des Datenbank-Skripts von Prof. Wegner spezifiziert → siehe Vorlesungshomepage

2

Uni-Schema



3

Professoren				Studenten			Vorlesungen			
PersNr	Name	Rang	Raum		Name	Semester	VorlNr	Titel	SWS	gelesen Von
2125	Sokrates	C4	226	24002	Xenokrates	18	5001	Grundzüge	4	2137
2126	Russel	C4	232	25403	Jonas	12	5041	Ethik	4	2125
2127	Kopernikus	C3	310	26120	Fichte	10	5043	Erkenntnistheorie	3	2126
2133	Popper	C3	52	26830	Aristoxenos	8	5049	Mäeutik	2	2125
2134	Augustinus	C3	309	27550	Schopenhauer	6	4052	Logik	4	2125
2136	Curie	C4	36	28106	Carnap	3	5052	Wissenschaftstheorie	3	2126
2137	Kant	C4	7	29120	Theophrastos	2	5216	Bioethik	2	2126
				29555	Feuerbach	2	5259	Der Wiener Kreis	2	2133
							5022	Glaube und Wissen	2	2134
							4630	Die 3 Kritiken	4	2137

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

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Assistenten			
PersNr	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

4

(Einfache) Datendefinition in SQL

Datentypen

- **character** (*n*), **char** (*n*)
- **character varying** (*n*), **varchar** (*n*)
- **numeric** (*p,s*), **integer**
- **blob** oder **raw** für sehr große binäre Daten
- **clob** für sehr große String-Attribute
- **date** für Datumsangaben

Anlegen von Tabelle

- **create table** Professoren
(PersNr **integer not null,**
Name **varchar (30) not null**
Rang **character (2));**

5

Veränderung am Datenbestand

Einfügen von Tupeln

insert into hören

```
select MatrNr, VorlNr
from Studenten, Vorlesungen
where Titel= `Logik` ;
```

insert into Studenten (MatrNr, Name)

```
values (28121, `Archimedes`);
```

6

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

Null-Wert

7

Veränderungen am Datenbestand

Löschen von Tupeln

delete Studenten

```
where Semester > 13;
```

Verändern von Tupeln

update Studenten

```
set Semester= Semester + 1;
```

8

Einfache SQL-Anfrage

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

PersNr	Name
2125	Sokrates
2126	Russel
2136	Curie
2137	Kant

Einfache SQL-Anfragen

Sortierung

select PersNr, Name, Rang
from Professoren
order by Rang **desc**, Name **asc**;

PersNr	Name	Rang
2136	Curie	C4
2137	Kant	C4
2126	Russel	C4
2125	Sokrates	C4
2134	Augustinus	C3
2127	Kopernikus	C3
2133	Popper	C3

Duplikateliminierung

select distinct Rang
from Professoren

Rang
C3
C4

- In der relationalen Algebra werden Duplikate automatisch eliminiert, aufgrund der Mengensemantik
- In SQL wird dies aus Effizienzgründen standardmäßig nicht gemacht.
- Auf Wunsch kann die Eliminierung mit „distinct“ erzwungen werden.

Professoren				Studenten			Vorlesungen			
PersNr	Name	Rang	Raum	MatrNr	Name	Semester	VorlNr	Titel	SWS	gelesen Von
2125	Sokrates	C4	226	24002	Xenokrates	18	5001	Grundzüge	4	2137
2126	Russel	C4	232	25403	Jonas	12	5041	Ethik	4	2125
2127	Kopernikus	C3	310	26120	Fichte	10	5043	Erkenntnistheorie	3	2126
2133	Popper	C3	52	26830	Aristoxenos	8	5049	Mäeutik	2	2125
2134	Augustinus	C3	309	27550	Schopenhauer	6	4052	Logik	4	2125
2136	Curie	C4	36	28106	Carnap	3	5052	Wissenschaftstheorie	3	2126
2137	Kant	C4	7	29120	Theophrastos	2	5216	Bioethik	2	2126
				29555	Feuerbach	2	5259	Der Wiener Kreis	2	2133
							5022	Glaube und Wissen	2	2134
							4630	Die 3 Kritiken	4	2137

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

prüfen	
MatrNr	VorlNr
28106	5001
25403	5041
27550	4630

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

Assistenten			
PersNr	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

Anfragen über mehrere Relationen

Welcher Professor liest "Mäeutik"?

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

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

Anfragen über mehrere Relationen

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

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

Verknüpfung X



PersN	Name	Rang	Raum	VorlNr	Titel	SWS	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

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

↓ Projektion

Name	Titel
Sokrates	Mäeutik

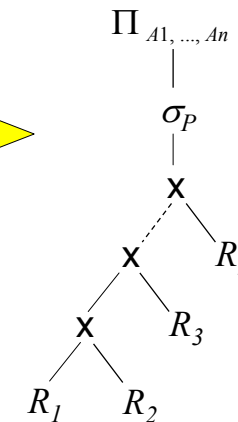
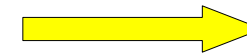
Kanonische Übersetzung in die relationale Algebra

Allgemein hat eine (ungeschachtelte) SQL-Anfrage die Form:

```
select A1, ..., An
from R1, ..., Rk
where P;
```

Übersetzung in die relationale Algebra:

$$\Pi_{A_1, \dots, A_n}(\sigma_P(R_1 \times \dots \times R_k))$$



Anfragen über mehrere Relationen

Welche Studenten hören welche Vorlesungen?

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

Alternativ:

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

17

Professoren				Studenten			Vorlesungen			
PersNr	Name	Rang	Raum		Name	Semester	VorlNr	Titel	SWS	gelesen Von
2125	Sokrates	C4	226	24002	Xenokrates	18	5001	Grundzüge	4	2137
2126	Russel	C4	232	25403	Jonas	12	5041	Ethik	4	2125
2127	Kopernikus	C3	310	26120	Fichte	10	5043	Erkenntnistheorie	3	2126
2133	Popper	C3	52	26830	Aristoxenos	8	5049	Mäeutik	2	2125
2134	Augustinus	C3	309	27550	Schopenhauer	6	4052	Logik	4	2125
2136	Curie	C4	36	28106	Carnap	3	5052	Wissenschaftstheorie	3	2126
2137	Kant	C4	7	29120	Theophrastos	2	5216	Bioethik	2	2126
				29555	Feuerbach	2	5259	Der Wiener Kreis	2	2133
							5022	Glaube und Wissen	2	2134
							4630	Die 3 Kritiken	4	2137

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

prüfen	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	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

18

Mengenoperationen und geschachtelte Anfragen

Mengenoperationen **union, intersect, minus**

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

19

Existenzquantor exists

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

20

Existenzquantor **exists**

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



21

Mengenvergleich

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

Unkorrelierte
Unterabfrage: meist
effizienter, wird nur
einmal ausgewertet

22

Der Vergleich mit "all"

Kein vollwertiger Allquantor!

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

23

Aggregatfunktion und Gruppierung

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

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

```
select gelesenVon, sum (SWS)  
from Vorlesungen  
group by gelesenVon;
```

```
select gelesenVon, Name, sum (SWS)  
from Vorlesungen, Professoren  
where gelesenVon = PersNr and Rang = 'C4'  
group by gelesenVon, Name  
having avg (SWS) >= 3;
```

24

Professoren				Studenten			Vorlesungen			
PersNr	Name	Rang	Raum		Name	Semester	VorlNr	Titel	SWS	gelesen Von
2125	Sokrates	C4	226	24002	Xenokrates	18	5001	Grundzüge	4	2137
2126	Russel	C4	232	25403	Jonas	12	5041	Ethik	4	2125
2127	Kopernikus	C3	310	26120	Fichte	10	5043	Erkenntnistheorie	3	2126
2133	Popper	C3	52	26830	Aristoxenos	8	5049	Mäeutik	2	2125
2134	Augustinus	C3	309	27550	Schopenhauer	6	4052	Logik	4	2125
2136	Curie	C4	36	28106	Carnap	3	5052	Wissenschaftstheorie	3	2126
2137	Kant	C4	7	29120	Theophrastos	2	5216	Bioethik	2	2126
				29555	Feuerbach	2	5259	Der Wiener Kreis	2	2133
							5022	Glaube und Wissen	2	2134
							4630	Die 3 Kritiken	4	2137

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

prüfen	
MatrNr	VorlNr
28106	5001
25403	5041
27550	4630

Assistenten			
PersNr	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

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

25

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

26

Ausführen einer Anfrage mit group by

```
select gelesenVon, Name, sum (SWS)
from Vorlesungen, Professoren
where gelesenVon = PersNr and Rang = 'C4'
group by gelesenVon, Name
having avg (SWS) >= 3;
```

Vorlesung x Professoren							
Vorl Nr	Titel	SWS	gelesen Von	PersNr	Name	Rang	Raum
5001	Grundzüge	4	2137	2125	Sokrates	C4	226
5041	Ethik	4	2125	2125	Sokrates	C4	226
...
4630	Die 3 Kritiken	4	2137	2137	Kant	C4	7

↓ where-Bedingung

27

```
select gelesenVon, Name, sum (SWS)
from Vorlesungen, Professoren
where gelesenVon = PersNr and Rang = 'C4'
group by gelesenVon, Name
having avg (SWS) >= 3;
```

VorlNr	Titel	SWS	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

28

VorINr	Titel	SWS	gelesenVon	PersNr	Name	Rang	Raum
5041	Ethik	4	2125	2125	Sokrates	C4	226
5049	Mäeutik	2	2125	2125	Sokrates	C4	226
4052	Logik	4	2125	2125	Sokrates	C4	226
5043	Erkenntnistheorie	3	2126	2126	Russel	C4	232
5052	Wissenschaftstheo.	3	2126	2126	Russel	C4	232
5216	Bioethik	2	2126	2126	Russel	C4	232
5001	Grundzüge	4	2137	2137	Kant	C4	7
4630	Die 3 Kritiken	4	2137	2137	Kant	C4	7

↓ **having**-Bedingung

VorINr	Titel	SWS	gelesenVon	PersNr	Name	Rang	Raum
5041	Ethik	4	2125	2125	Sokrates	C4	226
5049	Mäeutik	2	2125	2125	Sokrates	C4	226
4052	Logik	4	2125	2125	Sokrates	C4	226
5001	Grundzüge	4	2137	2137	Kant	C4	7
4630	Die 3 Kritiken	4	2137	2137	Kant	C4	7

↓ Aggregation (**sum**) und Projektion

29

```
select gelesenVon, Name, sum (SWS)
from Vorlesungen, Professoren
where gelesenVon = PersNr and Rang = 'C4'
group by gelesenVon, Name
having avg (SWS) >= 3;
```

gelesenVon	Name	sum (SWS)
2125	Sokrates	10
2137	Kant	8

30

Geschachtelte Anfrage (Forts.)

- Unteranfrage in der **where**-Klausel
- Welche Prüfungen sind besser als durchschnittlich verlaufen?

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

31

Geschachtelte Anfrage (Forts.)

- Unteranfrage in der **select**-Klausel
- Für jedes Ergebnistupel wird die Unteranfrage ausgeführt
- Man beachte, dass die Unteranfrage korreliert ist (greift auf Attribute der umschließenden Anfrage zu)

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

32

Professoren				Studenten			Vorlesungen			
PersNr	Name	Rang	Raum	Name	Semester	VorlNr	Titel	SWS	gelesen Von	
2125	Sokrates	C4	226	24002	Xenokrates	18	5001	Grundzüge	4	2137
2126	Russel	C4	232	25403	Jonas	12	5041	Ethik	4	2125
2127	Kopernikus	C3	310	26120	Fichte	10	5043	Erkenntnistheorie	3	2126
2133	Popper	C3	52	26830	Aristoxenos	8	5049	Mäeutik	2	2125
2134	Augustinus	C3	309	27550	Schopenhauer	6	4052	Logik	4	2125
2136	Curie	C4	36	28106	Carnap	3	5052	Wissenschaftstheorie	3	2126
2137	Kant	C4	7	29120	Theophrastos	2	5216	Bioethik	2	2126
				29555	Feuerbach	2	5259	Der Wiener Kreis	2	2133
							5022	Glaube und Wissen	2	2134
							4630	Die 3 Kritiken	4	2137

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

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	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

33

Unkorrelierte versus korrelierte Unteranfragen

- korrelierte Formulierung

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

34

- Äquivalente unkorrelierte Formulierung

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

- Vorteil: Unteranfrageergebnis kann materialisiert werden
- Unteranfrage braucht nur einmal ausgewertet zu werden

35

Entschachtelung korrelierter Unteranfragen -- Forts.

```
select a.*
from Assistenten a
where exists
(select p.*
from Professoren p
where a.Boss = p.PersNr and p.GebDatum>a.GebDatum);
```

- Entschachtelung durch Join

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

36

Verwertung der Ergebnismenge einer Unteranfrage

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

MatrNr	Name	VorlAnzahl
28106	Carnap	4
29120	Theophrastos	3

37

Decision-Support-Anfrage mit geschachtelten Unteranfragen

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

38

Das Ergebnis der Anfrage

VorlNr	AnzProVorl	GesamtAnz	Marktanteil
4052	1	8	.125
5001	4	8	.5
5022	2	8	.25
...

39

Weitere Anfragen mit Unteranfragen

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

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

40

```

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

```

41

Quantifizierte Anfragen in SQL

- Existenzquantor: **exists**

```

select Name
from Professoren
where not exists ( select *
                   from Vorlesungen
                   where gelesen Von = PersNr );

```

42

Allquantifizierung

- SQL-92 hat keinen Allquantor
- Allquantifizierung muß also durch eine äquivalente Anfrage mit Existenzquantifizierung ausgedrückt werden
- Kalkülformulierung der Anfrage: Wer hat **alle** vierstündigen Vorlesungen gehört?

$$\{s \mid s \in \text{Studenten} \wedge \forall v \in \text{Vorlesungen} (v.\text{SWS}=4 \Rightarrow \exists h \in \text{hören} (h.\text{VorlNr}=v.\text{VorlNr} \wedge h.\text{MatrNr}=s.\text{MatrNr}))\}$$

- Elimination von \forall und \Rightarrow
- Dazu sind folgende Äquivalenzen anzuwenden

$$\forall t \in R (P(t)) = \neg(\exists t \in R(\neg P(t)))$$

$$R \Rightarrow T = \neg R \vee T$$

43

Umformung des Kalkül-Ausdrucks ...

- Wir erhalten

$$\{s \mid s \in \text{Studenten} \wedge \neg(\exists v \in \text{Vorlesungen} \neg(\neg(v.\text{SWS}=4) \vee \exists h \in \text{hören} (h.\text{VorlNr}=v.\text{VorlNr} \wedge h.\text{MatrNr}=s.\text{MatrNr})))\}$$

- Anwendung von DeMorgan ergibt schließlich:

$$\{s \mid s \in \text{Studenten} \wedge \neg(\exists v \in \text{Vorlesungen} (v.\text{SWS}=4 \wedge \neg(\exists h \in \text{hören} (h.\text{VorlNr}=v.\text{VorlNr} \wedge h.\text{MatrNr}=s.\text{MatrNr}))))\}$$

44

- SQL-Umsetzung folgt direkt:

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

45

Allquantifizierung durch count-Aggregation

- Allquantifizierung kann immer auch durch eine **count**-Aggregation ausgedrückt werden
- Wir betrachten dazu eine etwas einfachere Anfrage, in der wir die (*MatrNr* der) Studenten ermitteln wollen, die *alle* Vorlesungen hören:

```
select h.MatrNr  
from hören h  
group by h.MatrNr  
  having count (*) = (select count (*) from Vorlesungen);
```

46

Herausforderung

- Wie formuliert man die komplexere Anfrage: Wer hat alle vierstündigen Vorlesungen gehört
- Grundidee besteht darin, vorher durch einen Join die Studenten/Vorlesungs-Paare einzuschränken und danach das Zählen durchzuführen

47

Nullwerte

- unbekannter Wert
- wird vielleicht später nachgereicht
- Nullwerte können auch im Zuge der Anfrageauswertung entstehen (Bsp. äußere Joins)
- manchmal sehr überraschende Anfrageergebnisse, wenn Nullwerte vorkommen

```
select count (*)  
from Studenten  
where Semester < 13 or Semester > =13
```

- Wenn es Studenten gibt, deren *Semester*-Attribut den Wert **null** hat, werden diese nicht mitgezählt
- Der Grund liegt in folgenden Regeln für den Umgang mit **null**-Werten begründet:

48

Auswertung bei Null-Werten

1. In arithmetischen Ausdrücken werden Nullwerte propagiert, d.h. sobald ein Operand **null** ist, wird auch das Ergebnis **null**. Dementsprechend wird z.B. **null** + 1 zu **null** ausgewertet- aber auch null * 0 wird zu **null** ausgewertet.
2. SQL hat eine dreiwertige Logik, die nicht nur **true** und **false** kennt, sondern auch einen dritten Wert **unknown**. Diesen Wert liefern Vergleichsoperationen zurück, wenn mindestens eines ihrer Argumente **null** ist. Beispielsweise wertet SQL das Prädikat (*PersNr=...*) immer zu **unknown** aus, wenn die *PersNr* des betreffenden Tupels den Wert **null** hat.
3. Logische Ausdrücke werden nach den folgenden Tabellen berechnet:

49

Diese Berechnungsvorschriften sind recht intuitiv. Unknown or true wird z.B. zu **true** - die Disjunktion ist mit dem **true**-Wert des rechten Arguments immer erfüllt, unabhängig von der Belegung des linken Arguments. Analog ist **unknown and false** automatisch **false** - keine Belegung des linken Arguments könnte die Konjunktion mehr erfüllen.

4. In einer **where**-Bedingung werden nur Tupel weitergereicht, für die die Bedingung **true** ist. Insbesondere werden Tupel, für die die Bedingung zu **unknown** ausgewertet, nicht ins Ergebnis aufgenommen.
5. Bei einer Gruppierung wird **null** als ein eigenständiger Wert aufgefaßt und in eine eigene Gruppe eingeordnet.

50

not	
true	false
unknown	unknown
false	true

and	true	unknown	false
true	true	unknown	false
unknown	unknown	unknown	false
false	false	false	false

or	true	unknown	false
true	true	true	true
unknown	true	unknown	unknown
false	true	unknown	false

51

Spezielle Sprachkonstrukte ("syntaktischer Zucker")

```

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);
    
```

52

```

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.VorINr = v.VorINr and
    v.Titel like `%thik%`;

```

53

Das case-Konstrukt

```

select MatrNr, ( case when Note < 1.5 then 'sehr gut'
                when Note < 2.5 then 'gut'
                when Note < 3.5 then 'befriedigend'
                when Note < 4.0 then 'ausreichend'
                else 'nicht bestanden' end)
from prüfen;

```

- Die **erste** qualifizierende **when**-Klausel wird ausgeführt

54

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 Name
from Vorlesungen v, hören h, Studenten s
where s.MatrNr = h.MatrNr and h.VorINr = v.VorINr and
    v.Titel = '%thik%';

```

55

Joins in SQL-92

- **cross join**: Kreuzprodukt
- **natural join**: natürlicher Join
- Join oder inner join: Theta-Join
- left, right oder full outer join: äußerer Join
- union join: Vereinigungs-Join (wird hier nicht vorgestellt)

```

select *
from  $R_1, R_2$ 
where  $= R_1.A = R_2.B;$ 

select *
from  $R_1$  join  $R_2$  on  $R_1.A = R_2.B;$ 

```

56

Äußere Joins

select p.PersNr, p.Name, f.PersNr, f.Note, f.MatrNr,
s.MatrNr, s.Name

from Professoren p **left outer join**

(prüfen f **left outer join** Studenten s **on** f.MatrNr=
s.MatrNr)

on p.PersNr=f.PersNr;

PersNr	p.Name	f.PersNr	f.Note	f.MatrNr	s.MatrNr	s.Name
2126	Russel	2126	1	28106	28106	Carnap
2125	Sokrates	2125	2	25403	25403	Jonas
2137	Kant	2137	2	27550	27550	Schopen- hauer
2136	Curie	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮

57

Äußere Joins

select p.PersNr, p.Name, f.PersNr, f.Note, f.MatrNr,
s.MatrNr, s.Name

from Professoren p **right outer join**

(prüfen f **right outer join** Studenten s **on**
f.MatrNr= s.MatrNr)

on p.PersNr=f.PersNr;

PersNr	p.Name	f.PersNr	f.Note	f.MatrNr	s.MatrNr	s.Name
2126	Russel	2126	1	28106	28106	Carnap
2125	Sokrates	2125	2	25403	25403	Jonas
2137	Kant	2137	2	27550	27550	Schopen- hauer
-	-	-	-	-	26120	Fichte
⋮	⋮	⋮	⋮	⋮	⋮	⋮

58

Äußere Joins

select p.PersNr, p.Name, f.PersNr, f.Note, f.MatrNr,
s.MatrNr, s.Name

from Professoren p **full outer join**

(prüfen f **full outer join** Studenten s **on**
f.MatrNr= s.MatrNr)

on p.PersNr=f.PersNr;

p.PersNr	p.Name	f.PersNr	f.Note	f.MatrNr	s.MatrNr	s.Name
2126	Russel	2126	1	28106	28106	Carnap
2125	Sokrates	2125	2	25403	25403	Jonas
2137	Kant	2137	2	27550	27550	Schopen- hauer
-	-	-	-	-	26120	Fichte
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2136	Curie	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮

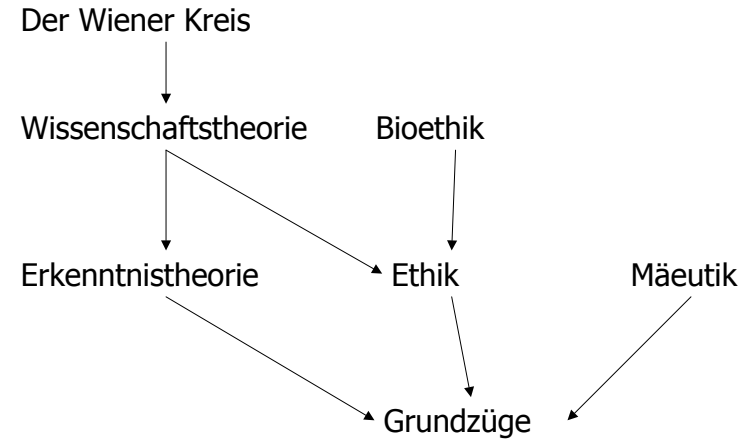
59

60

Rekursion

```
select Vorgänger
from voraussetzen, Vorlesungen
where Nachfolger= VorlNr and
        Titel= `Der Wiener Kreis`
```

61



62

Rekursion

```
select v1.Vorgänger
from voraussetzen v1, voraussetzen v2, Vorlesungen v
where v1.Nachfolger= v2.Vorgänger and
        v2.Nachfolger= v.VorlNr and
        v.Titel= `Der Wiener Kreis`
```

63

Vorgänger des „Wiener Kreises“ der Tiefe n

```
select v1.Vorgänger
from voraussetzen v1
        :
        voraussetzen vn_minus_1
        voraussetzen vn,
        Vorlesungen v
where v1.Nachfolger= v2.Vorgänger and
        :
        vn_minus_1.Nachfolger= vn.Vorgänger and
        vn.Nachfolger = v.VorlNr and
        v.Titel= `Der Wiener Kreis`
```

64

Transitive Hülle

$$trans_{A,B}(R) = \{(a,b) \mid \exists k \in \mathbb{N} (\exists t_1, \dots, t_k \in R ($$

$$t_1.A = t_2.B \wedge$$

$$\vdots$$

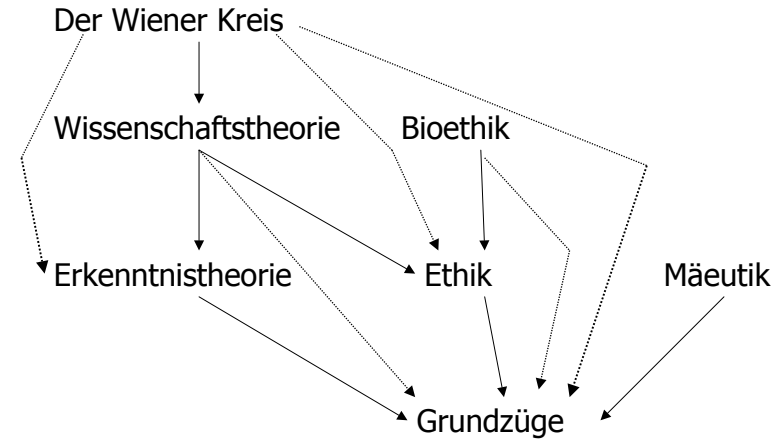
$$t_{k-1}.A = t_k.B \wedge$$

$$t_1.A = a \wedge$$

$$t_k.B = b))\}$$

Die transitive Hülle ist mit relationaler Algebra (und daher auch mit relationalem Kalkül) nicht zu realisieren. → Datalog

65



66

Die connect by-Klausel

```

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

67

Grundzüge
Ethik
Erkenntnistheorie
Wissenschaftstheorie

68

Rekursion in DB2/SQL99: gleiche Anfrage

with TransVorl (Vorg, Nachf)
as (**select** Vorgänger, Nachfolger **from** voraussetzen
union all
select t.Vorg, v.Nachfolger
from TransVorl t, voraussetzen v
where t.Nachf= v.Vorgänger)

select Titel **from** Vorlesungen **where** VorlNr **in**
(**select** Vorg **from** TransVorl **where** Nachf **in**
(**select** VorlNr **from** Vorlesungen
where Titel= `Der Wiener Kreis`))

69

- zuerst wird eine temporäre Sicht *TransVorl* mit der **with**-Klausel angelegt
- Diese Sicht *TransVorl* ist rekursiv definiert, da sie selbst in der Definition vorkommt
- Aus dieser Sicht werden dann die gewünschten Tupel extrahiert
- Ergebnis ist natürlich wie gehabt

70

Veränderung am Datenbestand

Einfügen von Tupeln

insert into hören
select MatrNr, VorlNr
from Studenten, Vorlesungen
where Titel= `Logik`;

insert into Studenten (MatrNr, Name)
values (28121, `Archimedes`);

71

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

72

Veränderungen am Datenbestand

Löschen von Tupeln

delete Studenten

where Semester > 13;

Verändern von Tupeln

update Studenten

set Semester= Semester + 1;

73

Zweistufiges Vorgehen bei Änderungen

1. die Kandidaten für die Änderung werden ermittelt und "markiert"
 2. die Änderung wird an den in Schritt 1. ermittelten Kandidaten durchgeführt
- Anderenfalls könnte die Änderungsoperation von der Reihenfolge der Tupel abhängen, wie folgendes Beispiel zeigt:

delete from voraussetzen

where Vorgänger **in** (**select** Nachfolger
from voraussetzen);

74

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5229

Ohne einen Markierungsschritt hängt das Ergebnis dieser Anfrage von der Reihenfolge der Tupel in der Relation ab. Eine Abarbeitung in der Reihenfolge der Beispielausprägung würde das letzte Tupel (5052, 5229) fälschlicherweise erhalten, da vorher bereits alle Tupel mit 5052 als *Nachfolger* entfernt wurden.

75

Sichten ...

für den Datenschutz

create view prüfenSicht as

select MatrNr, VorlNr, PersNr
from prüfen

76

Sichten ...

für die Vereinfachung von Anfragen

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

```
select distinct Semester  
from StudProf  
where PName= `Sokrates`;
```

77

Sichten zur Modellierung von Generalisierung

```
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);
```

78

```
create view Professoren as  
  select *  
  from Angestellte a, ProfDaten d  
  where a.PersNr=d.PersNr;
```

```
create view Assistenten as  
  select *  
  from Angestellte a, AssiDaten d  
  where a.PersNr=d.PersNr;
```

→ Untertypen als Sicht

79

```
create table Professoren  
  (PersNr integer not null,  
   Name varchar (30) not null,  
   Rang character (2),  
   Raum integer);
```

```
create table Assistenten  
  (PersNr integer not null,  
   Name varchar (30) not null,  
   Fachgebiet varchar (30),  
   Boss integer);
```

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

80

```

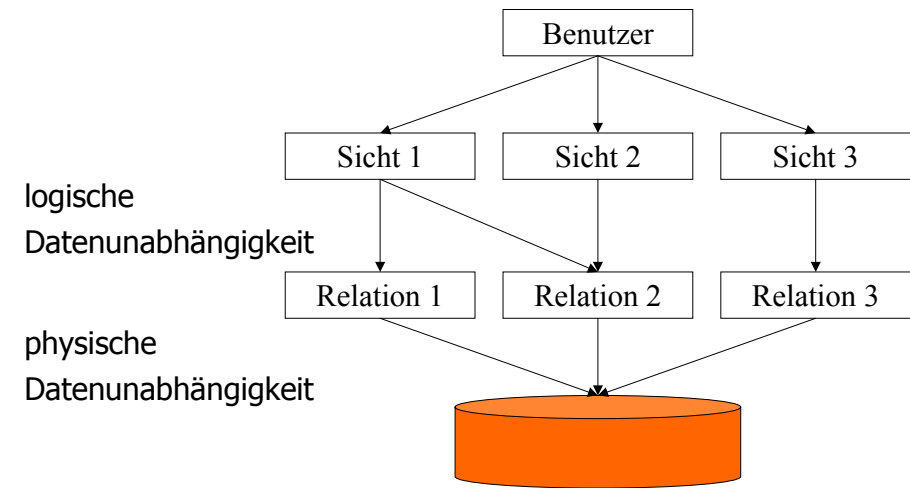
create view Angestellte as
  (select PersNr, Name
   from Professoren)
  union
  (select PersNr, Name
   from Assistenten)
  union
  (select*
   from AndereAngestellte);

```

→ Obertypen als Sicht

81

Sichten zur Gewährleistung von Datenunabhängigkeit



82

Änderbarkeit von Sichten

Beispiele für nicht änderbare Sichten

```

create view WieHartAlsPrüfer (PersNr, Durchschnittsnote) as
  select PersNr, avg(Note)
  from prüfen
  group by PersNr;

create view VorlesungenSicht as
  select Titel, SWS, Name
  from Vorlesungen, Professoren
  where gelesen Von=PersNr;

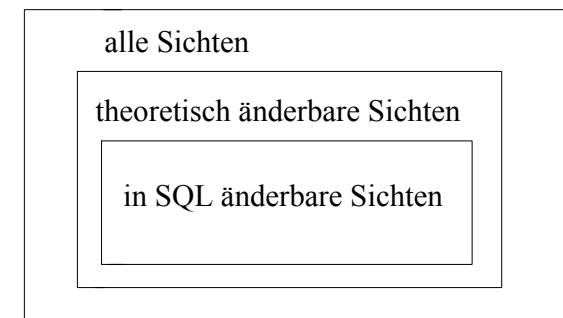
insert into VorlesungenSicht
  values (`Nihilismus`, 2, `Nobody`);

```

83

Änderbarkeit von Sichten

- in SQL
 - nur eine Basisrelation
 - Schlüssel muss vorhanden sein
 - keine Aggregatfunktionen, Gruppierung und Duplikateliminierung



84

Embedded SQL

```
#include <stdio.h>
/*Kommunikationsvariablen deklarieren */
exec sql begin declare section;
    varchar user_passwd[30];
    int exMatrNr;
exec sql end declare section;
exec sql include SQLCA;
main()
{
    printf("Name/Password:");
    scanf("%s", user_passwd.arr);
```

85

```
user_passwd.len=strlen(user_passwd.arr);
exec sql whenever sqlerror goto error;
exec sql connect :user_passwd;
while (1) {
    printf("Matrikelnummer (0 zum beenden):");
    scanf("%d", &exMatrNr);
    if (!exMatrNr) break;
    exec sql delete from Studenten
        where MatrNr= :exMatrNr;
}
exec sql commit work release;
exit(0);
```

86

```
error:
exec sql whenever sqlerror continue;
exec sql rollback work release;
printf("fehler aufgetreten!\n");
exit(-1);
}
```

87

Anfragen in Anwendungsprogrammen

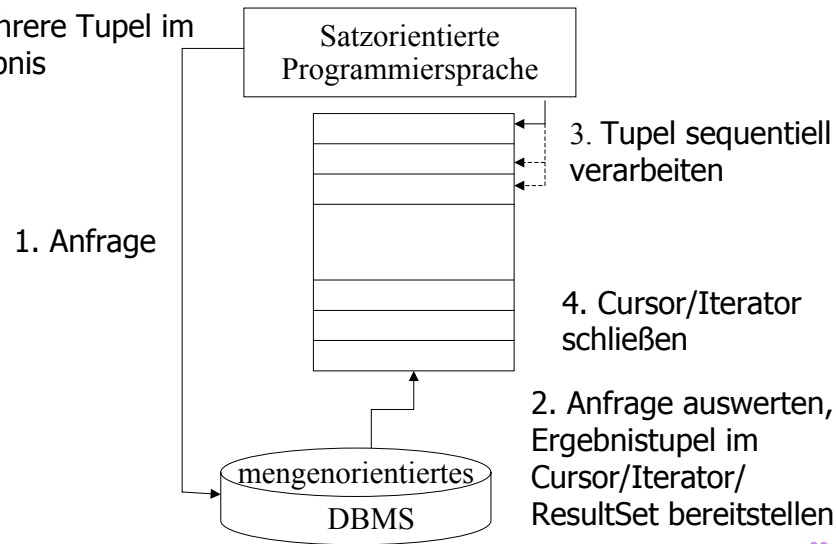
- genau ein Tupel im Ergebnis

```
exec sql select avg (Semester)
into :avgsem
from Studenten;
```

88

Anfragen in Anwendungsprogrammen

- mehrere Tupel im Ergebnis



89

Cursor-Schnittstelle in SQL

1. `exec sql declare c4profs cursor for select Name, Raum from Professoren where Rang='C4';`
2. `exec sql open c4profs;`
3. `exec sql fetch c4profs into :pname, :praum;`
4. `exec sql close c4profs;`

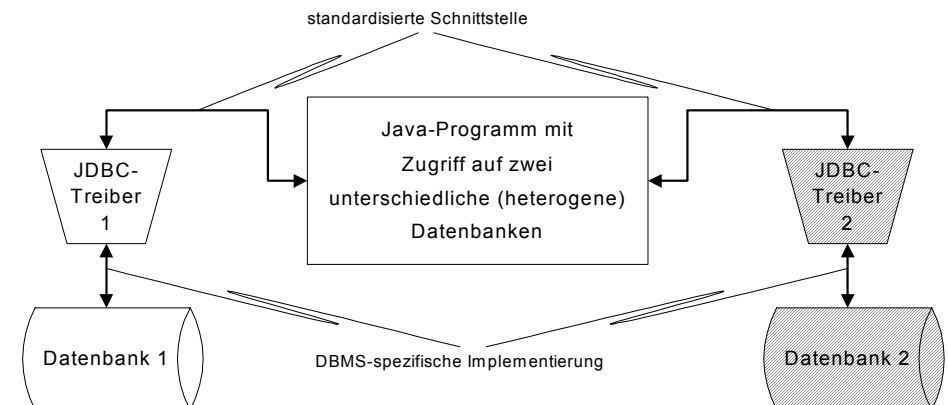
90

JDBC: Java Database Connectivity

- Standardisierte Schnittstelle zur Anbindung von relationalen Datenbanken an Java
- Wird heute fast immer für die Anbindung von Datenbanken an das Internet/Web verwendet
 - Java Servlets als dynamische Erweiterung von Webservern
 - Java Server Pages (JSP): HTML-Seiten mit eingebetteten Java Programmfragmenten

91

Zugriff auf Datenbanken via JDBC



92

Vorübersetzung von SQL-Ausdrücken

```
PreparedStatement sql_exmatrikuliere =
    conn.prepareStatement
        ("delete from Studenten where MatrNr = ?");

int VomBenutzerEingeleseneMatrNr;
// zu löschende MatrNr einlesen
sql_exmatrikuliere.setInt(1, VomBenutzerEingeleseneMatrNr);

int rows = sql_exmatrikuliere.executeUpdate();
if (rows == 1) System.out.println("StudentIn gelöscht.");
else System.out.println("Kein/e StudentIn mit dieser MatrNr.");
```

97

SQL/J-Beispielprogramm

```
import java.io.*; import java.sql.*;
import sqlj.runtime.*; import sqlj.runtime.ref.*;

#sql iterator StudentenItr (String Name, int Semester);

public class SQLJExmp {
    public static void main(String[] argv) {
        try {
            Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");
            Connection con = DriverManager.getConnection
                ("jdbc:db2:uni");

            con.setAutoCommit(false);
            DefaultContext ctx = new DefaultContext(con);
            DefaultContext.setDefaultContext(ctx);
```



```
StudentenItr Methusaleme;
#sql Methusaleme = { select s.Name, s.Semester
                    from Studenten s
                    where s.Semester > 13 };
while (Methusaleme.next()) {
    System.out.println(Methusaleme.Name() + ":" +
        Methusaleme.Semester());
}
Methusaleme.close();
#sql { delete from Studenten where Semester > 13 };
#sql { commit };
}
catch (SQLException e) {
    System.out.println("Fehler mit der DB-Verbindung: " + e);
}
catch (Exception e) {
    System.err.println("Folgender Fehler ist aufgetreten: " + e);
    System.exit(-1); } }
```

99

Query by Example

Vorlesungen	VorINr	Titel	SWS	gelesen Von
		p._t	> 3	

Analog

$$\{[t] \mid \exists v, s, r ([v,t,s,r] \in \text{Vorlesungen} \wedge s > 3)\}$$

Join in QBE

Vorlesungen	VorINr	Titel	SWS	gelesen Von
		Mäeutik		_x

Professoren	PersNr	Name	Rang	Raum
	_x	p._n		

100

Die Condition Box

Studenten	MatrNr	Name	Semester
		_s	_a

conditions
_a > _b

Studenten	MatrNr	Name	Semester
		_t	_b

Betreuen	potentieller Tutor	Betreuer
p.	_s	_t

Aggregatfunktion und Gruppierung

Vorlesungen	VorlNr	Titel	SWS	gelesen Von
			p.sum.all._x	p.g.

conditions
avg.all._x > 2

101

Updates in QBE: Sokrates ist „von uns gegangen“

Professoren	PersNr	Name	Rang	Raum
d.	_x	Sokrates		

Vorlesungen	VorlNr	Titel	SWS	gelesen Von
d.	_y			_x

hören	VorlNr	MatrNr
d.	_y	

102