

## from-Klausel

- Syntax

```
select *
from relationenliste
```
- Beispiel

```
select *
from Bcher
```

liefert die gesamte Relation Bcher

VL Datenbanken I – 8-3

VL Datenb.

## 8. Relationale Datenbanksprachen

- SQL-Kern
- Weitere Sprachkonstrukte von SQL
- SQL-Versionen
- Query by Example

## Kartesisches Produkt

- Bei mehr als einer Relation hinter **from**: kartesisches Produkt

```
select * from Bcher, Ausleih
```

Einführung von Tupelvariablen: etwa auf eine Relation mehrfach zugreifen

```
select * from Bcher eins, Bcher zwei
```
- Ergebnis hat acht Spalten:

```
eins.Invnr, eins.Titel,
eins.ISBN, eins.Autor,
zwei.Invnr, zwei.Titel,
zwei.ISBN, zwei.Autor
```
- **Selbst-Verbund (Self-Join)** für tupelübergreifende Selektionen

VL Datenbanken I – 8-4

VL Datenb.

## SQL-Kern

- select**
  - Projektionsliste
  - *arithmetische Operationen und Aggregatfunktionen*
- from**
  - zu verwendende Relationen, evtl. Umbenennungen
- where**
  - Selektions-, Verbundbedingungen
  - Geschachtelte Anfragen (wieder ein SFW-Block)
- group by**
  - *Gruppierung* für Aggregatfunktionen
- having**
  - *Selektionsbedingungen an Gruppen*

# Äußere Verbunde

Statt **inner join** nun **outer join** (dangling tuples übernehmen und mit Nullwerten auffüllen)

- **full outer join**: in beiden Operanden
- **left outer join**: im linken Operanden
- **right outer join**: im rechten Operanden

# SQL-92-Spezialitäten

- Verbunde als explizite Operatoren
- kartesisches Produkt  
`select * from Bcher, Ausleih`  
`select * from Bcher cross join Ausleih`
- Verbund über Verbundbedingungen  
`select * from Bcher, Ausleih`  
`where Bcher.Invnr = Ausleih.Invnr`
- **join-Operator**:  $\theta$ -Verbund  
`select *`  
`from Bcher join Ausleih`  
`on Bcher.Invnr = Ausleih.Invnr`

# Äußere Verbunde (II)

LINKS	<table border="1"><tr><th>A</th><th>B</th></tr><tr><td>1</td><td>2</td></tr><tr><td>2</td><td>3</td></tr></table>	A	B	1	2	2	3	RECHTS	<table border="1"><tr><th>B</th><th>C</th></tr><tr><td>3</td><td>4</td></tr><tr><td>4</td><td>5</td></tr></table>	B	C	3	4	4	5	NATURAL JOIN	<table border="1"><tr><th>A</th><th>B</th><th>C</th></tr><tr><td>2</td><td>3</td><td>4</td></tr></table>	A	B	C	2	3	4												
A	B																																		
1	2																																		
2	3																																		
B	C																																		
3	4																																		
4	5																																		
A	B	C																																	
2	3	4																																	
OUTER	<table border="1"><tr><th>A</th><th>B</th><th>C</th></tr><tr><td>1</td><td>2</td><td>⊥</td></tr><tr><td>2</td><td>3</td><td>4</td></tr><tr><td>⊥</td><td>4</td><td>5</td></tr></table>	A	B	C	1	2	⊥	2	3	4	⊥	4	5	LEFT	<table border="1"><tr><th>A</th><th>B</th><th>C</th></tr><tr><td>1</td><td>2</td><td>⊥</td></tr><tr><td>2</td><td>3</td><td>4</td></tr></table>	A	B	C	1	2	⊥	2	3	4	RIGHT	<table border="1"><tr><th>A</th><th>B</th><th>C</th></tr><tr><td>2</td><td>⊥</td><td>4</td></tr><tr><td>⊥</td><td>4</td><td>5</td></tr></table>	A	B	C	2	⊥	4	⊥	4	5
A	B	C																																	
1	2	⊥																																	
2	3	4																																	
⊥	4	5																																	
A	B	C																																	
1	2	⊥																																	
2	3	4																																	
A	B	C																																	
2	⊥	4																																	
⊥	4	5																																	

# Weitere Verbunde in SQL-92

- Gleichverbund  
`select * from Bcher join Ausleih`  
`using ( Inventarnr )`
- natürlicher Verbund  
`select * from Bcher natural join Ausleih`
- jeder SFW-Block hinter **from** (SQL-92 orthogonal)

## Tupelvariablen und Relationennamen

- Angabe der Attributnamen durch Präfix ergänzen

```
select ISBN from Bcher
```

und

```
select Bcher.ISBN from Bcher
```

- Tupelvariable kann benutzt werden:

```
select eins.ISBN, zwei.Titel  
from Bcher eins, Bcher zwei
```

VL Datenbanken I – 8-11

VL Daten

## Die select-Klausel

Relationenalgebra: abschließende Projektion  
Relationenkalkül: Zielliste

```
select [distinct] { attribut |  
  arithmetischer-ausdruck  
  | aggregat-funktion }*  
from ...
```

- Attribute aus **from**-Relationen
- Arithmetische Ausdrücke über Attributen und Konstanten
- Aggregatfunktionen über Attributen

**distinct**: Ergebnismenge statt Multimenge

VL Datenbanken I – 8-12

VL Daten

## Tupelvariablen und Relationennamen (II)

```
select ISBN, Titel, Stichwort (falsch!)  
from Bcher, Buch_Stichwort  
where Bcher.ISBN = Buch_Stichwort.ISBN
```

```
select Bcher.ISBN, Titel, Stichwort (richtig)  
from Bcher, Buch_Stichwort  
where Bcher.ISBN = Buch_Stichwort.ISBN
```

VL Datenbanken I – 8-12

VL Daten

## Projektion: Menge oder Multimenge

```
select Name from Ausleih
```

Name
Meyer
Schulz
Müller
Meyer

```
select distinct Name from Ausleih
```

Name
Meyer
Schulz
Müller

## Bereichsselektion

attribut **between** konstante1 **and** konstante2

Abkürzung für

attribut  $\geq$  konstante1 **and**  
attribut  $\leq$  konstante2

Beispiel

```
select Matrikelnummer from Prft  
where Note between 1.0 and 2.0
```

VL Datenbanken I – 8-15

VL Datenbanken

## Die where-Klausel

Selektionsbedingung der Relationenalgebra oder Verbundbedingung

```
select ... from ... where bedingung
```

Bedingung:

- **Konstanten-Selektion**  
attribut  $\theta$  konstante
- **Attribut-Selektion** zwischen Attributen mit kompatiblen Wertebereichen:  
attribut1  $\theta$  attribut2

## Ungewißheitsselektion

- theoretisch nur Abkürzung für disjunktiv verknüpfte Bedingung

- Syntax

attribut **like** spezialkonstante

- Spezialkonstante kann beinhalten
  - ◆ '%' (kein oder beliebig viele Zeichen)
  - ◆ '\_' (genau ein Zeichen)

VL Datenbanken I – 8-16

VL Datenbanken

## Verbundbedingung

```
relation1.attribut = relation2.attribut
```

Beispiel: natürlicher Verbund

```
select Bcher.Titel,  
       Bcher_Stichwort.Stichwort  
from Bcher, Buch_Stichwort  
where Bcher.ISBN = Buch_Stichwort.ISBN
```

auch Gleichverbund und  $\theta$ -Verbund erlaubt

## Schachtelung von Anfragen

- **where**-Klausel kann geschachtelt werden
- SFW-Blöcke liefern im allgemeinen mehrere Werte
- Vergleiche mit Wertemengen
  - ◆ Standardvergleiche in Verbindung mit Quantoren **all** ( $\forall$ ) oder **any** ( $\exists$ )
  - ◆ spezielle Prädikate für den Zugriff auf Mengen **in** und **exists**

VL Datenbanken I – 8-19

## Ungewiðheitsselektion (II)

- Anwendung: Selektion nach Büchern von Benjamin/Cummings

```
select * from Bcher
where Verlagsname like 'Benj%Cummings%'
```

ist Abkürzung für

```
select * from Bcher
where Verlagsname = 'Benjamin Cummings'
or Verlagsname = 'Benjamin/Cummings'
or Verlagsname = 'Benjamin-Cummings'
or Verlagsname = 'Benjamin and Cummin
or Verlagsname = 'BenjXFCummingsSCHLU
or ...
```

VL Datenbanken

## Das in-Prädikat

- Syntax:  
attribut **in** ( SFW-block )
- Beispiel:  

```
select Titel from Bcher
where ISBN in ( select ISBN from Empfiehlt )
```
- natürlicher Verbund mit nachfolgender Projektion

VL Datenbanken I – 8-20

## Weitere Bedingungen

- **Null-Selektion**  
attribut **is null**
- **Quantifizierte Bedingungen**, wenn ein Argument in Vergleich Menge liefert  
(**all**, **any**, **some** und **exists**)
- boolesche Ausdrücke mit Konnektoren  
**or**, **and** und **not**

VL Datenbanken

## Verzahnt geschachtelte Anfragen (II)

### ■ Abarbeitung

1. In der äußeren Anfrage das erste Personen-Tupel untersuchen  
Ergebnis in innere Anfrage einsetzen
2. innere Anfrage  

```
select Note
from Prft
where PANr = 4711
```

auswerten, liefert Werteliste ( 2.0, 2.3 )
3. Ergebnis der inneren Anfrage in die äußere einsetzen  
1.0 in ( 2.0, 2.3 ) ergibt **false**  
ersten Prüfer nicht berücksichtigen
4. in der äußeren Anfrage das zweite Personen-Tupel untersuchen usw.

VL Datenbanken I – 8-23

## Das in-Prädikat (II)

### ■ Abarbeitung

1. Ergebnis der inneren **select**-Anweisung hinter in a Liste von Konstanten einsetzen
2. Dann modifizierte Anfrage  

```
select Titel from Bcher
where ISBN in
( '3-929821-31-1', '0-201-53771-0',
'3-89319-175-5', '0-8053-1753-8'
)
abarbeiten
```

VL Datenbanken

## Das exists-Prädikat

- testet, ob Ergebnis der inneren Anfrage nicht leer

```
select ISBN
from Buch_Exemplare
where exists
( select *
  from Ausleihe
  where Invnr = Buch_Exemplare.Invnr )
```

VL Datenbanken I – 8-24

## Verzahnt geschachtelte Anfragen

- in der inneren Anfrage Relationen- oder Tupelvariablen-Name aus dem **from**-Teil der äußeren Anfrage verwenden

```
select Nachname
from Personen
where 1.0 in ( select Note
              from Prft
              where PANr = Personen.PA
```

VL Datenbanken

## Kompatible Attribute

- Attribute sind kompatibel bei kompatiblen Wertebereichen
- Zwei Wertebereiche sind kompatibel, wenn sie
  - ◆ gleich sind oder
  - ◆ beides auf `character` basierende Wertebereiche sind (unabhängig von der Länge der Strings) oder
  - ◆ beides numerische Wertebereiche sind (unabhängig von dem genauen Typ) wie `integer` oder `float`)
- Kompatible Attribute können in Vergleichen und Mengenoperationen benutzt werden

VL Datenbanken I – 8-27

## exists: Simulation des Allquantors

```
select Lehrstuhlbezeichnung
from Professoren
where not exists
  ( select * from Liest
    where Liest.PANr = Professoren.PANr
    and not exists ( select *
                    from Prft
                    where Prft.PANr =
                      Professoren.PANr
                    and Prft.V_Bezeichnung =
                      Liest.V_Bezeichnung ) )
```

VL Datenbanken

## SQL-89: Vereinigung

- SQL-89: Vereinigung **union** einzige Mengenoperation  
SFW\_block1 **union** SFW\_block2
- Beispiel:  

```
select A, B, C from R1
union
select A, C, D from R2
```

*Attributkompatibilität:* A von R1 und A von R2,  
B von R1 und C von R2, C von R1 und D von R2
- Ergebnis: Attributnamen des linken Operanden
- Vereinigung nur als „äußerste“ Operation erlaubt.

VL Datenbanken I – 8-28

## SQL-92: Tupelbildungen

- row constructors bilden Tupel aus Konstanten oder Attributen  $(e_1, \dots, e_n)$   

```
where ( select Studienfach, Immatdatum
        from Studenten
        where Matrikelnummer = 'HRO-91222
        = ('Informatik', '1.10.91') )
```
- Attribute müssen *kompatibel* sein (siehe unten)  
 $(a_1, \dots, a_n) < (b_1, \dots, b_n)$   
wahr, wenn ein  $j$  existiert, für das  $a_j < b_j$  und  $a_i = b_i$  für alle  $i < j$  gilt (lexikographische Ordnung)

VL Datenbanken

## Vereinigung, Durchschnitt und Differenz

- **SQL-92: union, intersect und except** orthogonal in andere Anfragen einsetzbar

```
select count(*)
from ( select PANr from Professoren)
union
      (select PANr from Studenten) )
```
- **corresponding-Klausel:** zwei Relationen nur über ihren gemeinsamen Bestandteilen vereinigen

```
select count(*)
from ( Professoren
      union corresponding Studenten) )
```

VL Datenbanken I – 8-31

## Simulation der Differenz

```
 $\pi[\text{PANr}] (\text{Mitarbeiter}) - \pi[\text{PANr}] (\text{Studenten})$ 
in SQL
select PANr from Mitarbeiter
where PANr not in ( select PANr
                   from Studenten )
```

VL Datenbanken I – 8-32

## Vergleich Relationenalgebra und SQL

Algebra	SQL-89	SQL-92
$\pi$	<b>select distinct</b>	<b>select distinct</b>
$\sigma$	<b>where</b> ohne Schachtelung	<b>where</b> ohne Schachtelung
$\bowtie$	<b>from, where</b>	<b>from, where</b>
$\beta$	<b>from</b> mit Tupelvariable	<b>from</b> mit <b>join</b> oder <b>natural join</b>
$-$	<b>where</b> mit Schachtelung	<b>from</b> mit Tupelvariable <b>as</b>
$\cap$	<b>where</b> mit Schachtelung	<b>where</b> mit Schachtelung <b>except corresponding</b>
$\cup$	<b>where</b> mit Schachtelung	<b>where</b> mit Schachtelung <b>intersect corresponding</b>
	<b>union</b> (nicht orthogonal)	<b>union corresponding</b>

VL Datenbanken I – 8-32

## Vereinigung und äußere Verbunde

```
select *
from Personen left outer natural join Pers_Tel
umgesetzt

select P.PANr, P.Vorname, P.Nachname, P.PLZ,
       P.Ort, P.Strae, P.HNr,
       P.Geburtsdatum, T.Telefon
from Personen P, Pers_Telefon T
where P.PANr = T.PANr
union
select P.PANr, P.Vorname, P.Nachname, P.PLZ,
       P.Ort, P.Strae, P.HNr, P.Geburtsdatum,
from Personen P
where not exists ( select *
                  from Pers_Telefon T
                  where P.PANr = T.PANr )
```

VL Datenbanken I – 8-32

## Operationen auf Wertebereichen (II)

- Ausdrücke werden tupelweise ausgewertet  
`select ISBN, Preis / 1.44  
from Buch_Versionen`

Ergebnis	ISBN	
	3-89319-175-5	54,86
	0-8053-1753-8	50,24
	0-8053-1753-8	61,70
	0-201-53771-0	60,73
	3-929821-31-1	54,86

VL Datenbanken I – 8-35

VL Datenbanken

## Weitere Sprachkonstrukte von SQL

- Operationen auf Wertebereichen
- Aggregatfunktionen
- **group by** und **having**
- Quantoren und Mengenvergleiche
- Beispiele für Selbst-Verbund
- **order by**
- Nullwerte
- Änderungs-Operationen

## SQL-92-Spezialitäten: Umbenennung

- im vorigen Beispiel: zweite Spalte nicht benannt
- in SQL-92: Attributname zuordnen  
`select ISBN, Preis / 1.44 as Dollar_Preis  
from Buch_Versionen`

VL Datenbanken I – 8-36

VL Datenbanken

## Operationen auf Wertebereichen

- innerhalb von **select** und **where**: statt Attribute auch *skalare Ausdrücke*
  - ◆ numerischen Wertebereiche: etwa +, -, \*, /
  - ◆ Strings: **char\_length**, Konkatenation ||, **substring** (Teilzeichenkette)
  - ◆ Datumsstypen, Zeitintervalle: **current\_date**, **current\_time**, +, -, \*

VL Datenbanken I – 8-36

VL Datenbanken

## Aggregatfunktionen: Beispiele

```
select sum(Preis) from Buch_Versionen
```

```
select count(*) from Professoren
```

```
select count(distinct PANr) from Prft
```

```
select avg(all Note) from Prft
where V_Bezeichnung = 'Datenbanken I'
```

↪ all notwendig

```
select Matrikelnr from Prft
where Note < ( select avg (all Note)
              from Prft )
```

VL Datenbanken I – 8-39

## Aggregatfunktionen

- built-in-Funktionen: tupelübergreifend
- ◆ **count**: Anzahl der Werte einer Spalte oder (Spezialfall **count(\*)**) Anzahl der Tupel einer Relation
- ◆ **sum**: Summe der Werte einer Spalte
- ◆ **avg**: arithmetisches Mittel der Werte einer Spalte
- ◆ **max** bzw. **min**: größter bzw. kleinster Wert einer Spalte
- Argumente einer Aggregatfunktion:
  - ◆ Attribut der durch **from** spezifizierten Relation
  - ◆ gültiger skalarer Ausdruck
  - ◆ bei **count** auch \*

VL Datenbanken I – 8-40

## group by und having

- Syntax

```
select ... from ... [ where ... ]
[ group by attributliste ]
[ having bedingung ]
```
- Semantik (virtuelle geschachtelte Relation):
  - ◆ Relationenschema  $R$  und Attributmenge hinter Gruppierung  $G$
  - ◆ schachteln nach Attributen  $R - G$ , d.h. für gleiche  $G$ -Werte werden Resttupel in Relation gesammelt
  - ◆ der **where**-Klausel genügende Tupel also schachteln gemäß

$$\nu[(R - G; N)](\tau(R))$$

VL Datenbanken I – 8-40

## Aggregatfunktionen (II)

- Vor Argument (außer bei **count(\*)**) optional: **distinct** oder **all** (**all** Voreinstellung)
- Nullwerte werden vor Anwendung aus Wertemenge eliminiert (außer bei **count(\*)**)

VL Datenbanken I – 8-40

## Gruppierung: Schritt 1

- from und where

A	B	C	D
1	2	3	4
1	2	4	5
2	3	3	4
3	3	4	5
3	3	6	7
...			



A	B	C	D
1	2	3	4
1	2	4	5
2	3	3	4
3	3	4	5
3	3	6	7

## group by und having II

- having ist Selektionsbedingung auf gruppierter Relation
- darf Bezug nehmen auf
  - ◆ Gruppierungsattribute
  - ◆ beliebige Aggregatfunktionen über Nicht-Gruppierungsattributen

## Gruppierung: Schritt 2

- group by A, B

A	B	C	D
1	2	3	4
1	2	4	5
2	3	3	4
3	3	4	5
3	3	6	7



A	B	N	
		C	D
1	2	3	4
		4	5
2	3	3	4
3	3	4	5
		6	7

## Gruppierung: Schema

- Relation REL:

A	B	C	D
1	2	3	4
1	2	4	5
2	3	3	4
3	3	4	5
3	3	6	7
...			

- Anfrage:

`select A, sum(D) from REL where ...  
group by A, B  
having A < 4 and sum(D) < 10 and max(C) = 4`

## Gruppierung: Beispiele

```
select count(*) as Anzahl, PANr
from Ausleihe
group by PANr
```

Anzahl	PANr
2	7754
1	4711
1	5588
2	9912

VL Datenbanken I – 8-47

## Gruppierung: Beispiele (II)

```
select count(*), PANr from Ausleihe
group by PANr
having count(*) > 1
```

	PANr
2	7754
2	9912

```
select Matrikelnummer from Prft
group by Matrikelnummer
having avg(Note) < (select avg(Note)
from Prft)
```

VL Datenbanken I – 8-48

## Gruppierung: Schritt 3

```
■ select A, sum(D)
```

A	B	N	C	D
1	2	3	4	4
		4	5	
2	3	3	4	4
3	3	4	5	5
		6	7	



A	sum(D)	N	
		C	D
1	9	3	4
		4	5
2	4	3	4
3	12	4	5
		6	7

VL Datenbanken I – 8-47

## Gruppierung: Schritt 4

```
■ having A < 4 and sum(D) < 10 and max(C) = 4
```

A	sum(D)	N	
		C	D
1	9	3	4
		4	5
2	4	3	4
3	12	4	5
		6	7



A	sum(D)
1	9

VL Datenbanken I – 8-48

# Selbst-Verbund

- letzte Anfrage erst mit Selbst-Verbund zu lösen
- Vergleich von Wertemengen

```
select BA_1.ISBN
from Buch_Autor BA_1, Buch_Autor BA_2
where BA_1.ISBN = BA_2.ISBN
and BA_1.Autor = 'Vossen'
and BA_2.Autor = 'Witt'
```

BA_1.ISBN	BA_1.Autor	BA_2.ISBN	BA_2.Autor
3-89319-175-5	Vossen	3-89319-175-5	Vossen
3-89319-175-5	Vossen	3-89319-175-5	Witt
3-89319-175-5	Vossen	0-8053-1753-8	Elmasri
...			
3-89319-175-5	Witt	3-89319-175-5	Vossen
...			

VL Datenbanken I – 8-51

# Quantoren und Mengenvergleiche

- Syntax  
attribut  $\theta$   
`all | any | some` } ( `select` attribut  
from ... `where` ... )
- Bedeutung all Allquantor, any, some Existenzquantor
- Beispiele  
`select` PANr, Immatrulationsdatum  
from Studenten  
where Matrikelnummer = any ( `select` Matrikelnummer from Prft

VL Datenbanken I – 8-52

# Selbst-Verbund (II)

- Zählen von Wertemengen  
`select distinct` X.PANr  
from Prft X, Prft Y  
where X.PANr = Y.PANr  
and X.Matrikelnummer <>  
Y.Matrikelnummer

VL Datenbanken I – 8-52

# Quantoren und Mengenvergleiche (II)

```
select Note from Prft
where Matrikelnummer = 'HRO-912291'
and Note >= all (
select Note from Prft
where Matrikelnummer = 'HRO-912291
```

Anwendbarkeit eingeschränkt: Test auf Mengen-Gleichheit

$$\forall x \in M_1 : \exists x \in M_2 \wedge \forall x \in M_2 : \exists x \in M_1$$

in SQL so nicht umsetzbar:

*Gib alle Bücher aus, an denen 'Vossen' und 'Witt' gemeinsam als Autoren beteiligt waren*

VL Datenbanken I – 8-52

## Behandlung von Nullwerten

- skalare Ausdrücke: Ergebnis **null**, sobald Nullwert in die Berechnung eingeht
- In allen Aggregatfunktionen bis auf **count(\*)** werden Nullwerte vor Anwendung der Funktion entfernt
- Fast alle Vergleiche mit Nullwert ergeben Wahrheitswert **unknown** (statt **true** oder **false**). Ausnahme: **is null** ergibt **true**, **is not null** ergibt **false**
- Boolesche Ausdrücke basieren dann auf dreiwertiger Logik

VL Datenbanken I – 8-55

## order by-Klausel

- Menge von Tupeln  $\leadsto$  Liste
- Syntax  
order by attributListe
- Beispiel  
select Matrikelnummer, Note  
from Prft  
where V\_Bezeichnung = 'Datenbanken I',  
order by Note **asc**
- aufsteigend (**asc**) oder absteigend (**desc**) sortieren

VL Datenbanken I – 8-56

## Behandlung von Nullwerten (II)

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

not	
true	false
unknown	unknown
false	true

VL Datenbanken I – 8-56

## order by-Klausel (II)

- Sortierung wird auf das Ergebnis der jeweils vorangehenden SFW-Anfrage angewendet, also FALSCH:  
select Matrikelnummer  
from Prft  
where V\_Bezeichnung = 'Datenbanken I',  
order by Note (falsch!)

VL Datenbanken I – 8-56

## update (II)

Angestellte

Name	Gehalt
Meyer	4000
Bond	7200
Schulz	5400

```
update Angestellte set Gehalt = 6000
where Name = 'Bond'

update Angestellte set Gehalt = 3000
```

VL Datenbanken I – 8-59

## Änderungsoperationen

- Einfügen von Tupeln in Basisrelationen (oder Sichten)  
**insert**
- Löschen von Tupeln aus Basisrelationen (oder Sichten)  
**delete**
- Ändern von Tupeln in Basisrelationen (oder Sichten)  
**update**

diese Operationen jeweils als

- Eintupel-Operationen (etwa die Erfassung einer neuer Ausleiher)
- Mehrtupel-Operationen (erhöhe das Gehalt aller Mitarbeiter um 4.5%)

SQL: vor allem Mehrtupel-Operationen

VL Datenbanken

## delete

- Syntax  
**delete from** basisrelation  
[ **where** bedingung ]
- Beispiel  
**delete from** Ausleihe **where** Invnr = 4711
- Standardfall ist Löschen mehrerer Tupel:  
**delete from** Ausleihe **where** Name = 'Meyer'
- Löschen der gesamten Relation:  
**delete from** Ausleihe

VL Datenbanken I – 8-60

## update

- Syntax  
**update** basisrelation  
**set** attribut\_1 = ausdr\_1, ...,  
attribut\_n = ausdr\_n  
[ **where** bedingung ]
- Beispiele

Angestellte	Name	Gehalt
	Meyer	3000
	Bond	7200
	Schulz	4400

```
update Angestellte set Gehalt=Gehalt+1
where Gehalt < 5000
```

VL Datenbanken

## SQL-89

- Level 1
  - ◆ keine Nullwerte
  - ◆ keine Selektionsbedingungen mit  $\neq$  oder **exists**
  - ◆ keine **union**-Operation
  - ◆ ...
- Level 2: wie hier beschrieben
- Level 2 + IEF (Integrity Enhancement Feature)
  - ◆ **check**-Klausel: **where**-Klausel als Integritätsbedingung
  - ◆ Definition von Primärschlüsseln und Fremdschlüsseln

VL Datenbanken I – 8-63

## insert

```
insert into basisrelation
[ (attribut_1, ..., attribut_n) ]
values (konstante_1, ..., konstante_n)

insert into Buch (Invnr, Titel)
values (4867, 'Wissensbanken')

insert into Buch
values (4867, 'Wissensbanken', '3-87', 'Karaj

insert into basisrelation
[ (attribut_1, ..., attribut_n) ]
SQL-anfrage

insert into Kunde (select LName, LAdr, 0
from Lieferant )
```

VL Datenbanken

## SQL-92

- neue Datentypen (wie `interval`)
- Domänenkonzept (**create domain**, **alter domain**)
- Änderung des Datenbankschemas: **alter table** und **drop table**
- allgemeine Integritätsbedingungen (mehrere Tabellen)
- `string`-Operationen erweitert
- Namen für abgeleitete Spalten
- **join**, **cross join**, **natural join**, **outer join** als eigene Operatoren
- auch **intersect** und **except**

VL Datenbanken I – 8-64

## SQL-Versionen

- Geschichte
  - ◆ SEQUEL (1974, IBM Research Labs San Jose)
  - ◆ SEQUEL2 (1976, IBM Research Labs San Jose)
  - ◆ SQL (1982, IBM)
  - ◆ ANSI-SQL (SQL-86; 1986)
  - ◆ ISO-SQL (SQL-89; 1989; drei Sprachen Level 1, Level 2, + IEF)
  - ◆ (ANSI / ISO) SQL2 (als SQL-92 verabschiedet)
  - ◆ (ANSI / ISO) SQL3 (als SQL:1999 verabschiedet)
  - ◆ (ANSI / ISO) SQL:2003 (in Arbeit)

VL Datenbanken

## SQL-92 (IV)

Feature in SQL-92	Entry	Intermediate	Full
<b>corresponding</b> bei Mengenoperationen	-	-	+
dreiwertige Logik	-	-	+
allgemeine Integritätsbedingungen	-	-	+
<b>check</b> mit Bezug zu anderen Tabellen	-	-	+
<b>alter domain</b>	-	-	+
Tabellenkonstruktoren	-	-	+

VL Datenbanken I – 8-67

## Die Sprache QBE

- „Query by Example“
- Anfragen in QBE: Einträge in Tabellengerüsten
- Intuition: *Beispieleinträge* in Tabellen
- Vorläufer verschiedener tabellenbasierter  
Anfrageschnittstellen kommerzieller Systeme

VL Datenbanken I – 8-68

## SQL-92 (II)

- Sprache fast vollständig orthogonal (etwa **union**, SFW hinter **from**)
- dreiwertige Logik
- **set transaction**: verschiedene Isolationsstufen
- Embedded SQL und Dynamic SQL sind Teil der Norm (siehe Abschnitt „Anwendungsprogrammierung“)
- Data Dictionary ist Teil der Norm

VL Datenbanken I – 8-65

## SQL-92 (III)

Feature in SQL-92	Entry	Intermediate	Full
Datum, Intervalltypen, <b>domain</b>	-	+	+
string-Operationen	-	+	+
<b>join</b>	-	+	+
<b>except, intersect</b>	-	+	+
<b>alter, drop table</b>	-	+	+
<b>set transaction</b>	-	+	+
Dynamic SQL	-	+	+
<b>union</b> orthogonal	-	+	+
andere Orthogonalitäts- verbesserungen	-	+	+

VL Datenbanken I – 8-66

## Anfragen in QBE III

„Welche Vorlesungen werden für die Studiengänge Informatik und Mathematik angeboten?“

Vorl	V_Bez	SWS	Semester	Studiengang
P.	P.			_St

CONDITIONS

\_St = Informatik or \_St = Mathematik

$\{xy \mid \exists s \text{ Vorl}(x, y, \_, s) \wedge (s = \text{'Informatik'} \vee s = \text{'Mathematik'})\}$

VL Datenbanken I – 8-71

## Anfragen in QBE

„Alle für das Fach Informatik angebotenen Vorlesungen ab dem 7ten Semester.“

Vorl	V_Bez	SWS	Semester	Studiengang
P.	P.	P.	> 7	Informatik

$\{xy \mid \text{Vorl}(x, y, z, \text{'Informatik'}) \wedge z > 7\}$

VL Datenbanken I – 8-71

## Alternative Lösung

„Welche Vorlesungen werden für die Studiengänge Informatik und Mathematik angeboten?“

Vorl	V_Bez	SWS	Semester	Studiengang
P.	_DBA	P.	_eins	Informatik
P.	_DBB	P.	_zwei	Mathematik

$\{xy \mid \exists s \text{ Vorl}(x, y, \_, s) \wedge (s = \text{'Informatik'} \vee s = \text{'Mathematik'})\}$

VL Datenbanken I – 8-72

## Anfragen in QBE II

„Für welche Vorlesungen mit mehr als 2 Semesterwochenstunden ist 'Datenbanken I' Voraussetzung?“

Vorl	V_Bez	SWS	Semester	Studiengang
P.	_DB	P.	> 2	

Vorl_Voraus	V_Bez	Voraussetzung
	_DB	Datenbanken I

$\{xy \mid \text{Vorl}(x, y, \_, \_) \wedge \text{Vorl\_Voraus}(x, \text{'Datenbanken I'}) \wedge y$

VL Datenbanken I – 8-72

## Anfragen in QBE VI

„Drucke Vorlesungen mit maximaler Anzahl von Semesterwochenstunden.“

Vorl	V_Bez	SWS	Semester	Studiengang
P.		_viele		
⊖		> _viele		

$\{xyzw \mid \text{Vorl}(x, y, z, w) \wedge \neg \exists x' \exists y' \exists z' \exists w' (\text{Vorl}(x', y', z', w') \wedge y' > y)\}$

VL Datenbanken I – 8-75

## Sortierung in QBE

Vorl	V_Bez	SWS	Semester	Studiengang
P.			AO(2).	AO(1).

VL Datenbanken I – 8-76

## Anfragen in QBE IV

„Welche Vorlesungen werden in einem Fach im selben Semester gehört?“

Vorl	V_Bez	SWS	Semester	Studiengang
P.			_erstes	_Informatik
P.			_erstes	P. _Informatik

$\{xyz \mid \text{Vorl}(x, \_, w, z) \wedge \text{Vorl}(y, \_, w, z)\}$

VL Datenbanken I – 8-74

## Anfragen in QBE V

„Drucke alle Vorlesungen für Informatik mit den Voraussetzungen!“

Vorl	V_Bez	SWS	Semester	Studiengang
	_DB	_zwei	_erstes	Informatik
	Vorl_Voraus	V_Bez	Voraussetzung	
		_DB	_DBVoraus	

Info_VL	Name	Voraussetzung	SWS	Semester
P.	_DB	_DBVoraus	_zwei	_erstes

$\{xyz \mid \text{Vorl}(x, y, z, \text{'Informatik'}) \wedge \text{Vorl\_Voraus}(x, w)\}$

VL Datenbanken I – 8-73

# Formale Semantik von QBE

- Beispiелеlemente von QBE entsprechen Bereichsvariablen des Bereichskalküls
- Analog ‘\_’-Symbolen im Kalkül entsprechen “leere Spalteneinträge” paarweise verschiedenen Bereichsvariablen. leere Spalten: alle Positionen, an denen keine Bereichsvariable steht (auch <10)
- Jede Zeile in einer Relation  $R$  entspricht Teilformel

$$R(u_1, \dots, u_n) \wedge \phi$$

wobei  $u_i$  Konstanten oder Bereichsvariablen und  $\phi$  Konjunktion der jeweiligen Zeilenbedingungen

# Formale Semantik von QBE II

- allgemeine QBE-Anfrage (positive, negierte Zeilen, Einträge in der Condition Box) entspricht dem folgendem Bereichskalkül-Ausdruck:

$$\{x_1 \dots x_m \mid \exists y_1 \dots \exists y_n \bigwedge_i \langle i\text{-te positive Zeile} \rangle \wedge \bigwedge_j \neg [\exists z_{j_1} \dots \exists z_{j_p} \langle j\text{-te negierte Zeile} \rangle] \wedge \bigwedge_k \langle k\text{-te Bedingung in Condition Box} \rangle\}$$

- ◆  $x_1 \dots x_m$  Variablen mit  $P$ .
- ◆  $y_1 \dots y_n$  restliche Variablen in positiven Zeilen
- ◆  $z_{j_1} \dots z_{j_p}$  restliche implizite Variablen in der  $j$ -ten negierten Zeile

# Aggregation in QBE

Vorl	V_Bez	SWS	Semester	Studiengang
		P.SUM.ALL.		Informatik
		_x		

# Aggregation in QBE II

Buch_Autor	ISBN	Autor
		P.CNT.UN.ALL.
		_x

# Änderungen in QBE

„Einfügen eines konkreten Kunden.“

KUNDE	KName	KAdr	Kto
I.	Dagobert Duck	Entenhausen	1.000.000

# Ausdrucksfähigkeit von QBE

QBE-Anfragen können nur Kalkülausdrücken der folgenden Form entsprechen:

$$\{ \dots \mid \exists \dots (\wedge \dots \neg (\exists \dots \exists \dots) \wedge \dots) \}$$

$$\{ \dots \mid \exists \dots \forall \dots \forall \dots (\text{Rest ohne Quantoren}) \}$$

Der QBE-Kern ist relational vollständig.

# Änderungen in QBE II

„Einfügen aller Lieferanten als Kunden.“

KUNDE	KName	KAdr	Kto
I.	_Mller	_Ort	0

  

LIEFERANT	LName	LAdr	Ware	Preis
	_Mller	_Ort		

# Relationenalgebra und QBE

Algebra	QBE
Projektion	mit P. markierte Spalten
Selektion	1. Vergleiche als Spalteneinträge 2. Condition Box
Umbenennung	explizite Ausgabetablelle
Verbund	Verbindung zweier Tabellen mittels Beispiелеlementen (Bereichsvarianten)

## Änderungen in QBE V

Alternative Formulierung:

KUNDE	KName	KAdr	Kto
U.	Meier	_MD	_AlterWert
	Meier	_MD	_NeuerWert
CONDITIONS			
_NeuerWert = _AlterWert - 110			

VL Datenbanken I – 8-87

## Änderungen in QBE III

„Löschen aller schlechten Kunden.“

KUNDE	KName	KAdr	Kto
D.			<0

VL Datenbanken I – 8-87

## QBE in MS-Access

- MS-Access: Datenbankprogramm für Windows
  - ◆ Basisrelationen mit Schlüssel
  - ◆ Fremdschlüssel über graphische Angabe von Beziehungen
  - ◆ graphische Definition von Anfragen (SQL-ähnlich)
  - ◆ interaktive Definition von Formularen und Berichten
- Unterstützung von QBE

VL Datenbanken I – 8-88

## Änderungen in QBE IV

„110.- DM von Meiers Konto abziehen.“

KUNDE	KName	KAdr	Kto
	Meier		U._Wert - 110

Verkürzte Form der folgenden Änderung:

KUNDE	KName	KAdr	Kto
U.	Meier	_MD	_Wert
	Meier	_MD	_Wert - 110

VL Datenbanken I – 8-88



# Access: Verbund

The screenshot shows the Microsoft Access interface for a query named 'Auswahlabfrage'. The table design grid is visible, showing fields: V\_Bezeichnung, Studiengang, SWS, and Semester. A criteria table is also shown below the design grid.

Vorlesungen	Vorl_Voraus
* V_Bezeichnung Studiengang SWS Semester	* V_Bezeichnung Voraussetzung

  

Feld:	V_Bezeichnung	SWS	Voraussetzung
Tabelle:	Vorlesungen	Vorlesungen	Vorl_Voraus
Sortierung:			
Anzeigen:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Kriterien:		>2	"Datenbanken I"
oder:			