

Kapitel 4: Relationale Datenbanksprachen

Join (Natural Join) (1)

Einleitung
SQL-Kern
Weitere
Konstrukte
Änderungen
Schluß

Ausleih

| Invnr | Name |
|-------|--------|
| 4711 | Meyer |
| 1201 | Schulz |
| 0007 | Müller |
| 4712 | Meyer |

Bücher

| Invnr | ... | Autor |
|-------|-----|------------|
| 0007 | ... | James Bond |
| 1201 | ... | Heuer |
| 4711 | ... | Vossen |
| 4712 | ... | Ullman |
| 4717 | ... | Wirth |

| Invnr | Name | ... | Autor |
|-------|--------|-----|------------|
| 0007 | Müller | ... | James Bond |
| 1201 | Schulz | ... | Heuer |
| 4711 | Meyer | ... | Vossen |
| 4712 | Meyer | ... | Ullman |

Join (Natural Join) (2)

Einleitung
SQL-Kern
Weitere
Konstrukte
Änderungen
Schluß

Ausleih

| Invnr | ISBN | Name |
|-------|------|--------|
| 4711 | 1234 | Meyer |
| 1201 | 5678 | Schulz |
| 0007 | 9012 | Müller |
| 4712 | 3456 | Meyer |

Bücher

| Invnr | ISBN | Autor |
|-------|------|------------|
| 0007 | 9012 | James Bond |
| 1201 | 5678 | Heuer |
| 4711 | 1234 | Vossen |
| 4712 | 4713 | Ullman |
| 4717 | 1234 | Wirth |

- Was ist das Join-Ergebnis?
 - ◆ Welches Schema?
 - ◆ Welche Tupel?

Join (Natural Join) (3)

Einleitung
SQL-Kern
Weitere
Konstrukte
Änderungen
Schluß

Ausleih

| Invnr | Name |
|-------|--------|
| 4711 | Meyer |
| 1201 | Schulz |
| 0007 | Müller |
| 4712 | Meyer |

Bücher

| ISBN | Autor |
|------|------------|
| 9012 | James Bond |
| 5678 | Heuer |
| 1234 | Vossen |
| 4713 | Ullman |
| 1234 | Wirth |

| Invnr | Name | ISBN | Autor |
|-------|--------|------|------------|
| 4711 | Meyer | 9012 | James Bond |
| 4711 | Meyer | 5678 | Heuer |
| 4711 | Meyer | 1234 | Vossen |
| ... | ... | ... | ... |
| 1201 | Schulz | 9012 | James Bond |
| 1201 | Schulz | 5678 | Heuer |
| ... | ... | ... | ... |

Kartesisches
Produkt

Aggregatfunktionen (1)

- Prinzip: Berechnung eines Werts aus allen (oder zumindest) Werten eines Attributs.
- Aggregatfunktionen in Standard SQL: COUNT(), SUM(), MIN(), MAX(), AVG()
- SQL-Erweiterungen beinhalten zusätzliche Aggregatfunktionen (Statistik, Physik).

Aggregatfunktionen (2)

| Ausleih | Invnr | ISBN | Name |
|---------|-------|------|--------|
| | 4711 | 1234 | Meyer |
| | 1201 | 5678 | Schulz |
| | 0007 | 9012 | Müller |
| | 4712 | 3456 | Meyer |

- count(Invnr)=4,
- count(Name)=3

Group-by Operator (1)

| Marke | Datum | Bundesland | Anzahl |
|-------|------------|------------|--------|
| BMW | 07.01.1994 | Hessen | 28 |
| BMW | 08.01.1994 | Bayern | 37 |
| BMW | 07.01.1994 | Saarland | 41 |
| Opel | 07.01.1994 | Hessen | 48 |
| Opel | 08.01.1994 | Bayern | 62 |
| Opel | 08.01.1994 | Saarland | 5 |
| Opel | 09.01.1994 | Saarland | 95 |
| Audi | 07.01.1994 | Hessen | 55 |
| Audi | 08.01.1994 | Bayern | 52 |
| Audi | 09.01.1994 | Bayern | 27 |
| Audi | 10.01.1994 | Bayern | 62 |

| Marke | Anzahl |
|-------|--------|
| BMW | 106 |
| Opel | 210 |
| Audi | 196 |

Marke → sum

| Marke | Anzahl |
|-------|--------|
| BMW | 35.33 |
| Opel | 52.5 |
| Audi | 49 |

Marke → avg

| Marke | Anzahl |
|-------|--------|
| BMW | 41 |
| Opel | 95 |
| Audi | 62 |

Marke → max

| Bundesland | Anzahl |
|------------|--------|
| Hessen | 131 |
| Bayern | 240 |
| Saarland | 141 |

Bundesland → sum

Group-by Operator (2)

| Marke | Datum | Bundesland | Anzahl |
|-------|------------|------------|--------|
| BMW | 07.01.1994 | Hessen | 28 |
| BMW | 08.01.1994 | Bayern | 37 |
| BMW | 07.01.1994 | Saarland | 41 |
| Opel | 07.01.1994 | Hessen | 48 |
| Opel | 08.01.1994 | Bayern | 62 |
| Opel | 08.01.1994 | Saarland | 5 |
| Opel | 09.01.1994 | Saarland | 95 |
| Audi | 07.01.1994 | Hessen | 55 |
| Audi | 08.01.1994 | Bayern | 52 |
| Audi | 09.01.1994 | Bayern | 27 |
| Audi | 10.01.1994 | Bayern | 62 |

| Marke | Bundesland | Anzahl |
|-------|------------|--------|
| BMW | Hessen | 28 |
| BMW | Bayern | 37 |
| BMW | Saarland | 41 |
| Opel | Hessen | 48 |
| Opel | Bayern | 62 |
| Opel | Saarland | 100 |
| Audi | Hessen | 55 |
| Audi | Bayern | 141 |

Marke, Bundesland → sum

Relationale Datenbanksprachen

- SQL-Kern,
- Weitere Sprachkonstrukte von SQL,
- SQL-Versionen.

SQL-Kern

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

from-Klausel

- Syntax:
select *
from relationenliste
- Beispiel:
select *
from Bücher
liefert die gesamte Relation Bücher.

Laufendes Beispiel

| Ausleih | Invnr | Name |
|---------|-------|--------|
| | 4711 | Meyer |
| | 1201 | Schulz |
| | 0007 | Müller |
| | 4712 | Meyer |

| Bücher | Invnr | Titel | ISBN | Autor |
|--------|-------|--------------|-------|------------|
| | 0007 | Dr. No | 3-125 | James Bond |
| | 1201 | Objektbanken | 3-111 | Heuer |
| | 4711 | Datenbanken | 3-765 | Vossen |
| | 4712 | Datenbanken | 3-891 | Ullman |
| | 4717 | Pascal | 3-999 | Wirth |

Kartesisches Produkt

- Bei mehr als einer Relation hinter **from**: Kartesisches Produkt.

```
select * from Bücher, Ausleih
```

Self-Join – Illustration (1)

- Zugrundeliegende Relation:

| Person | NAME | VORNAME |
|--------|-----------|---------|
| | Böhm | Klemens |
| | Buchmann | Erik |
| | Duckstein | Ralf |
| | Saake | Gunter |

Attributnamen
 kommen
 mehrmals vor!

- Gewünscht: Alle Personen-Paare:

| NAME | VORNAME | NAME | VORNAME |
|----------|---------|----------|---------|
| Böhm | Klemens | Böhm | Klemens |
| Böhm | Klemens | Buchmann | Erik |
| ... | ... | ... | ... |
| Buchmann | Erik | Böhm | Klemens |
| ... | ... | ... | ... |

- *Self-Join*:

Kartesisches Produkt einer Tabelle mit sich selbst.

Self-Join – Illustration (2)

- Einführung von Tupelvariablen:
 Etwa auf eine Relation mehrfach zugreifen.

```
select * from Person eins, Person zwei  

where eins.Alter < zwei.Alter
```

- Ergebnis hat vier Spalten:
eins.Name, eins.Vorname,
zwei.Name, zwei.Vorname

- *Selbst-Verbund* (Self-Join)
 für tupelübergreifende Selektionen.

- Weiteres Beispiel:

```
select * from Person eins, Person zwei  

where eins.Alter < zwei.Alter
```

Zugrundeliegende Relation für das folgende Beispiel

| Bücher2 | ISBN | TITEL | VNAME |
|---------|-------|----------------------------|----------|
| | 12345 | Datenbanken | Springer |
| | 23456 | Weltalmanach | Fischer |
| | 34567 | Der Oberst hat niemand ... | Suhrkamp |

Verbindung zum Datenbanksserver erfolgreich
Bitte Anfrage eingeben:

```
select * from Bücher2 eins, Bücher2 zwei
```

Abfrage starten

```
select * from Bücher2 eins, Bücher2 zwei
```

| ISBN | TITEL | VNAME | ISBN | TITEL | |
|-------|----------------------------|----------|-------|----------------------------|----------|
| 12345 | Datenbanken | Springer | 12345 | Datenbanken | Springer |
| 23456 | Weltalmanach | Fischer | 12345 | Datenbanken | Springer |
| 34567 | Der Oberst hat niemand ... | Suhrkamp | 12345 | Datenbanken | Springer |
| 12345 | Datenbanken | Springer | 23456 | Weltalmanach | Fischer |
| 23456 | Weltalmanach | Fischer | 23456 | Weltalmanach | Fischer |
| 34567 | Der Oberst hat niemand ... | Suhrkamp | 23456 | Weltalmanach | Fischer |
| 12345 | Datenbanken | Springer | 34567 | Der Oberst hat niemand ... | Suhrkamp |
| 23456 | Weltalmanach | Fischer | 34567 | Der Oberst hat niemand ... | Suhrkamp |
| 34567 | Der Oberst hat niemand ... | Suhrkamp | 34567 | Der Oberst hat niemand ... | Suhrkamp |

| time | percent | rows | query |
|----------|---------|------|--|
| 0.003518 | 100% | 9 | select * from Bücher2 eins, Bücher2 zwei |
| 0.003518 | | | |

Einleitung
SQL-Kern
- Übersicht
- from
- select
- where
- Mengenop.
Weitere
Konstrukte
Änderungen
Schluß

SQL-92 spezielle Aspekte

- Verbunde als explizite Operatoren.
- Kartesisches Produkt:
`select * from Bücher, Ausleih`
`select * from Bücher cross join Ausleih`
- Verbund über Verbundbedingungen:
`select * from Bücher, Ausleih`
`where Bücher.Invnr = Ausleih.Invnr`
- join-Operator: θ -Verbund
`select *`
`from Bücher join Ausleih`
`on Bücher.Invnr = Ausleih.Invnr`

Theta Join – Beispiel

| PROF | | DKE | |
|---------------|----------|---------|-----------|
| PName | PVorname | Vorname | Name |
| Saake | Gunter | Klemens | Böhm |
| Rautenstrauch | Claus | Erik | Buchmann |
| Böhm | Klemens | Ralf | Duckstein |

PROF \bowtie PVorname < Vorname DKE

| PName | PVorname | Vorname | Nachname |
|---------------|----------|---------|-----------|
| Saake | Gunter | Klemens | Böhm |
| Saake | Gunter | Ralf | Duckstein |
| Rautenstrauch | Claus | Klemens | Böhm |
| Rautenstrauch | Claus | Erik | Buchmann |
| Rautenstrauch | Claus | Ralf | Duckstein |
| Böhm | Klemens | Ralf | Duckstein |

Statt < auch \geq , \neq , \approx usw. möglich.

Einleitung
SQL-Kern
- Übersicht
- from
- select
- where
- Mengenop.
Weitere
Konstrukte
Änderungen
Schluß

Weitere Verbunde in SQL-92

- Gleichverbund:
`select * from Bücher join Ausleih`
`using (Inventarnr)`
- natürlicher Verbund:
`select * from Bücher natural join Ausleih`
Besser als herkömmliche Formulierung, weil
 - ◆ übersichtlicher,
 - ◆ weniger fehleranfällig
(z. B. vergißt man leicht ein Attribut, wenn man alle explizit aufzählen muß).
- Jeder SFW-Block kann in SQL-92 auch hinter **from** Klausel eingesetzt werden (Orthogonalität). Nicht mit SQL-89.

Einleitung
SQL-Kern
- Übersicht
- from
- select
- where
- Mengenop.
Weitere
Konstrukte
Änderungen
Schluß

Äußere Verbunde (1)

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.
- **Beispiel**:

```
select * from Verlage right outer join
Bücher2 using(VName)
```

Äußere Verbunde (2)

| | | | | | | | | | |
|-------|---|---|--------|---|---|-----------------|---|---|---|
| LINKS | A | B | RECHTS | B | C | NATURAL JOIN | A | B | C |
| | 1 | 2 | | 3 | 4 | | 2 | 3 | 4 |
| | 2 | 3 | | 4 | 5 | | | | |

| | | | | | | | | | | | |
|-------|---|---|---|------|---|---|---|-------|---|---|---|
| OUTER | A | B | C | LEFT | A | B | C | RIGHT | A | B | C |
| | 1 | 2 | ⊥ | | 1 | 2 | ⊥ | | 2 | 3 | 4 |
| | 2 | 3 | 4 | | 2 | 3 | 4 | | ⊥ | 4 | 5 |
| | ⊥ | 4 | 5 | | | | | | | | |

Die select-Klausel

- Abschließende Projektion, Zielliste.
- **Syntax**:

```
select [distinct] { attribut |
    arithmetischer-ausdruck |
    aggregat-funktion }
from ...
```
- **Bestandteile**:
 - ◆ Attribute aus **from**-Relationen,
 - ◆ arithmetische Ausdrücke über Attributen und Konstanten,
 - ◆ Aggregatfunktionen über Attributen.
- **distinct**: Ergebnismenge statt Multimenge.

Laufendes Beispiel

| | | |
|---------|-------|--------|
| Ausleih | Invnr | Name |
| | 4711 | Meyer |
| | 1201 | Schulz |
| | 0007 | Müller |
| | 4712 | Meyer |

| | | | | |
|--------|-------|--------------|-------|------------|
| Bücher | Invnr | Titel | ISBN | Autor |
| | 0007 | Dr. No | 3-125 | James Bond |
| | 1201 | Objektbanken | 3-111 | Heuer |
| | 4711 | Datenbanken | 3-765 | Vossen |
| | 4712 | Datenbanken | 3-891 | Ullman |
| | 4717 | Pascal | 3-999 | Wirth |

Projektion: Menge oder Multimenge

- `select Name from Ausleih`

| |
|--------|
| Name |
| Meyer |
| Schulz |
| Müller |
| Meyer |

- `select distinct Name from Ausleih`

| |
|--------|
| Name |
| Meyer |
| Schulz |
| Müller |

Tupelvariablen und Relationennamen (1)

- Attributnamen durch Präfix ergänzen:

```
select ISBN from Bücher
```

und

```
select Bücher.ISBN from Bücher
```

- Tupelvariable kann benutzt werden:

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

Tupelvariablen und Relationennamen (2)

```
select ISBN, Titel, Stichwort falsch  
from Bücher, Buch_Stichwort  
where Bücher.ISBN = Buch_Stichwort.ISBN
```

```
select Bücher.ISBN, Titel, Stichwort richtig  
from Bücher, Buch_Stichwort  
where Bücher.ISBN = Buch_Stichwort.ISBN
```

Die where-Klausel

- Selektionsbedingung oder Verbundbedingung.
- Syntax:

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

- Bedingung:

- ◆ *Konstanten-Selektion:*

```
attribut  $\theta$  konstante
```

- ◆ *Attribut-Selektion* zwischen Attributen
mit kompatiblen Wertebereichen:

```
attribut1  $\theta$  attribut2
```

Konstantenselektion

- Beispiel:
select *
from AUSLEIH
where NAME = 'Meyer'

Verbundbedingung

- `relation1.attribut = relation2.attribut`
- Beispiel: natürlicher Verbund
select Bücher.Titel,
 Bücher_Stichwort.Stichwort
from Bücher, Buch_Stichwort
where Bücher.ISBN = Buch_Stichwort.ISBN
- Auch Gleichverbund und θ -Verbund erlaubt.
(*Gleichverbund* – Attribute, die nicht gleich heißen,
mit `,=` vergleichen.)

Bereichsselektion

- `attribut between konstante1 and konstante2`
- Abkürzung für
`attribut \geq konstante1 and`
`attribut \leq konstante2`
- Beispiel:
select Matrikelnummer **from** Prüft
where Note **between** 1.0 **and** 2.0

Ungewißheitsselektion (1)

- Theoretisch nur Abkürzung
für disjunktiv verknüpfte Bedingung.
- Syntax:
`attribut like spezialkonstante`
- Spezialkonstante kann beinhalten
 - ◆ `'%'` – kein oder beliebig viele Zeichen,
 - ◆ `'_'` – genau ein Zeichen.

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

Ungewißheitsselektion (2)

- Anwendung: Selektion nach Büchern von Benjamin/Cummings

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

ist Abkürzung für

```
select * from Bücher
where Verlagsname = 'Benjamin Cummings'
or Verlagsname = 'Benjamin/Cummings'
or Verlagsname = 'Benjamin-Cummings'
or Verlagsname =
    'Benjamin and Cummings'
or Verlagsname =
    'BenjXFCummingsSCHlumpf'
or ...
```

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

Weitere Bedingungen

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

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

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**.

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

Das in-Prädikat (1)

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

```
select Titel from Bücher
where ISBN in (select ISBN from Empfiehlt)
```

| | | | | | | |
|--------|------|-------|--------|-----------|------|------|
| Bücher | ISBN | Titel | Verlag | Empfiehlt | PANr | ISBN |
|--------|------|-------|--------|-----------|------|------|

Einleitung
SQL-Kern
- Übersicht
- from
- select
- where
- Mengenop.
Weitere
Konstrukte
Änderungen
Schluß

Das in-Prädikat (2)

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

```
select Titel from Bücher
where ISBN in
( '3-929821-31-1', '0-201-53771-0',
  '3-89319-175-5' , '0-8053-1753-8' )
abearbeiten.
```
- Wie sieht äquivalente Anfrage ohne Schachtelung aus?

```
select Titel from Bücher
where ISBN in (select ISBN from Empfiehlt)
```

Einleitung
SQL-Kern
- Übersicht
- from
- select
- where
- Mengenop.
Weitere
Konstrukte
Änderungen
Schluß

Verzahnt geschachtelte Anfragen (1)

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

- Beispiel:

| Personen | PANr | Nachname | Büro |
|----------|------|----------|------|
| | | | |

| Prüft | PANr | Matrikel | Vorlesung | Ort | Zeit | Note |
|-------|------|----------|-----------|-----|------|------|
| | | | | | | |

```
select Nachname
from Personen
where 1.0 in (select Note
              from Prüft
              where PANr = Personen.PANr)
```

Einleitung
SQL-Kern
- Übersicht
- from
- select
- where
- Mengenop.
Weitere
Konstrukte
Änderungen
Schluß

Verzahnt geschachtelte Anfragen (2)

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

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

auswerten, liefert Werteliste (2.0, 2.3).

Einleitung
SQL-Kern
- Übersicht
- from
- select
- where
- Mengenop.
Weitere
Konstrukte
Änderungen
Schluß

Verzahnt geschachtelte Anfragen (3)

- Abarbeitung (Fortsetzung):
 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.
- Was leistet Anfrage also, anschaulich gesprochen?

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

Verzahnt geschachtelte Anfragen (4)

| | | | | | | |
|----------|------|----------|-----------|-----|------|------|
| Personen | PANr | Nachname | Büro | | | |
| Prüft | PANr | Matrikel | Vorlesung | Ort | Zeit | Note |

- **select** Nachname
from Personen
where 1.0 **in** (**select** Note
from Prüft
where PANr = Personen.PANr)
- Wie sieht äquivalente Anfrage ohne Schachtelung aus?

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

Das exists-Prädikat (1)

- Testet, ob Ergebnis der inneren Anfrage nicht leer.

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

| | | | | |
|----------------|-------|--------------|---------------|-------|
| Buch_Exemplare | Invnr | ISBN | Standort | |
| Ausleihe | Invnr | Ausleiher-Nr | Ausleih-Datum | Frist |

- Was leistet die Anfrage (anschaulich)?
 Als nicht geschachtelte Anfrage darstellbar?

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

Das exists-Prädikat (2)

- Formulieren Sie Anfrage „Selektiere alle Bücher, von denen alle Exemplare präsent sind.“.

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

exists: Simulation des Allquantors (1)

- Beispiel – drei zugrundeliegende Relationen:

| | | | | | |
|-------|---------------|-------|---------------|-------------|----------------------|
| Prüft | | Liest | | Professoren | |
| PANr | V_Bezeichnung | PANr | V_Bezeichnung | PANr | Lehrstuhlbezeichnung |

- Gesucht: „Alle Professoren, die für alle Vorlesungen, die sie lesen, eine Prüfung abnehmen.“

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

Alle Vorlesungen,
 die ein bestimmter Prof.
 hält, aber nicht prüft.

exists: Simulation des Allquantors (2)

- Beispiel:

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

- Drei zugrundeliegende Relationen:

| Prüft | | Liest | | Professoren | |
|-------|---------------|-------|---------------|-------------|----------------------|
| PANr | V_Bezeichnung | PANr | V_Bezeichnung | PANr | Lehrstuhlbezeichnung |
| | | | | | |

Kompatible Attribute (1)

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

- Attribute sind kompatibel bei kompatiblen Wertebereichen.
- Zwei Wertebereiche sind kompatibel, wenn sie
 - ◆ gleich sind,
 - ◆ beide auf character basierende Wertebereiche sind (unabhängig von der Länge der Strings), oder
 - ◆ beide numerische Wertebereiche sind (unabhängig vom genauen Typ) wie integer oder float.

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

Kompatible Attribute (2)

- Kompatible Attribute können in Vergleichen und Mengenoperationen benutzt werden.
- Beispiel: '... where Salary < 50000', und Salary ist vom Typ float.

SQL-92: Tupelbildungen (1)

Einleitung
 SQL-Kern
 - Übersicht
 - from
 - select
 - where
 - Mengenop.
 Weitere
 Konstrukte
 Änderungen
 Schluß

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

```
where (select Studienfach, Immatdatum
      from Studenten
      where Matrikelnummer = 'HRO-912291')
=
('Informatik', '1.10.91')
```

SQL-92: Tupelbildungen (2)

- $(a_1, \dots, a_n) < (b_1, \dots, b_n)$
 - wahr, wenn ein j existiert, für das $a_j < b_j$ und $a_i = b_i$ für alle $i < j$ gilt (lexikographische Ordnung).
 - Beispiel: (Böhm, Klemens, Magdeburg) < (Böhm, Klemens, Zürich)
- Attribute müssen *kompatibel* sein.
Beispiel: (Böhm, Klemens, 39114)
nicht vergleichbar mit (Böhm, Klemens, Zürich)

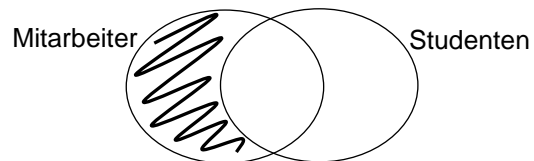
SQL-89: Vereinigung

- SQL-89: Vereinigung **union**
einzige Mengenoperation:
`SFW_block1 union SFW_block2`
- *Positionsweise Vereinigung.*
- 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.

Simulation der Differenz



- In SQL:

```
select PANr from Mitarbeiter
where PANr not in (select PANr
                  from Studenten)
```
- Gibt es hierzu äquivalente,
nicht geschachtelte Anfrage?

Vereinigung und äußere Verbunde

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

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

Vereinigung, Durchschnitt und Differenz

- SQL-92: **union**, **intersect** und **except** (Oracle: **minus**)
orthogonal in andere Anfragen einsetzbar.

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

- **corresponding**-Klausel: Zwei Relationen nur über ihre gemeinsamen Attribute vereinigen.

```
select count(*)  
from (Professoren  
union corresponding Studenten))
```
- D. h. alle anderen Bestandteile fallen weg.

Z

Mögliche Prüfungsfragen

- Formulieren diverser Anfragen, auch komplexerer Natur, in SQL.
- Vorgegebene geschachtelte Anfrage als nicht-geschachtelte Anfrage hinschreiben.
- Welche Join-Varianten kennen Sie?
- Geben Sie ein Beispiel, in dem ein Self-Join sinnvoll ist.
- Was ist der Zusammenhang zwischen Vereinigung und Outer Join?

Weitere Sprachkonstrukte von SQL

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

Operationen auf Wertebereichen (1)

- Innerhalb von **select** und **where**:
Statt Attributen auch *skalare Ausdrücke*.
 - ◆ Numerische Wertebereiche: etwa + , - , * , /,
 - ◆ Strings: **char_length**, Konkatenation .. , **substring** (Teilzeichenkette),
 - ◆ Datumstypen, Zeitintervalle: **current_date**, **current_time**, + , - , *.

Einleitung
SQL-Kern
Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
Änderungen
Schluß

Operationen auf Wertebereichen (2)

- 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 |

Einleitung
SQL-Kern
Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
Änderungen
Schluß

SQL-92-Spezialitäten: Umbenennung

- Im vorigen Beispiel: Zweite Spalte nicht benannt.
- In SQL-92: Attributnamen zuordnen.

```
select ISBN, Preis / 1.44 as Dollar_Preis  
from Buch_Versionen
```

Ergebnis

| ISBN | Dollar_Preis |
|---------------|--------------|
| 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 |

Einleitung
SQL-Kern
Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
Änderungen
Schluß

Aggregatfunktionen: Beispiele

- `select sum(all Preis) from Buch_Versionen`
- `select sum(distinct Preis) from Buch_Versionen`
- `select sum(Preis) from Buch_Versionen`
- `select count(*) from Professoren`
- `select count(distinct PANr) from Prüft`
- `select avg(all Note) from Prüft`
`where V_Bezeichnung = 'Datenbanken I'`
- `select Matrikelnummer from Prüft`
`where Note < (select avg(all Note)`
`from Prüft)`

Einleitung
SQL-Kern
Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
Änderungen
Schluß

Aggregatfunktionen (1)

- 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 *****.

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

Aggregatfunktionen (2)

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

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

Group-by Operator (1)

| Marke | Datum | Bundesland | Anzahl |
|-------|-------------|------------|--------|
| BMW | 07.01. 1994 | Hessen | 28 |
| BMW | 08.01. 1994 | Bayern | 37 |
| BMW | 07.01. 1994 | Saarland | 41 |
| Opel | 07.01. 1994 | Hessen | 48 |
| Opel | 08.01. 1994 | Bayern | 62 |
| Opel | 08.01. 1994 | Saarland | 5 |
| Opel | 09.01. 1994 | Saarland | 95 |
| Audi | 07.01. 1994 | Hessen | 55 |
| Audi | 08.01. 1994 | Bayern | 52 |
| Audi | 09.01. 1994 | Bayern | 27 |
| Audi | 10.01. 1994 | Bayern | 62 |

| Marke | Anzahl |
|-------|--------|
| BMW | 106 |
| Opel | 210 |
| Audi | 196 |

sum

| Marke | Anzahl |
|-------|--------|
| BMW | 35.33 |
| Opel | 52.5 |
| Audi | 49 |

avg

| Marke | Anzahl |
|-------|--------|
| BMW | 41 |
| Opel | 95 |
| Audi | 62 |

max

| Bundesland | Anzahl |
|------------|--------|
| Hessen | 131 |
| Bayern | 240 |
| Saarland | 141 |

sum

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

Group-by Operator (2)

| Marke | Datum | Bundesland | Anzahl |
|-------|-------------|------------|--------|
| BMW | 07.01. 1994 | Hessen | 28 |
| BMW | 08.01. 1994 | Bayern | 37 |
| BMW | 07.01. 1994 | Saarland | 41 |
| Opel | 07.01. 1994 | Hessen | 48 |
| Opel | 08.01. 1994 | Bayern | 62 |
| Opel | 08.01. 1994 | Saarland | 5 |
| Opel | 09.01. 1994 | Saarland | 95 |
| Audi | 07.01. 1994 | Hessen | 55 |
| Audi | 08.01. 1994 | Bayern | 52 |
| Audi | 09.01. 1994 | Bayern | 27 |
| Audi | 10.01. 1994 | Bayern | 62 |

| Marke | Bundesland | Anzahl |
|-------|------------|--------|
| BMW | Hessen | 28 |
| BMW | Bayern | 37 |
| BMW | Saarland | 41 |
| Opel | Hessen | 48 |
| Opel | Bayern | 62 |
| Opel | Saarland | 100 |
| Audi | Hessen | 55 |
| Audi | Bayern | 141 |

Marke, Bundesland
sum

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

group by und having (1)

- 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.

group by und having (2)

- Beispiele von eben:
 - ◆ **select** Marke, sum(Anzahl)
from Zulassungen
group by Marke
 - ◆ **select** Marke, avg(all Anzahl)
from Zulassungen
group by Marke
 - ◆ **select** Marke, max(Anzahl)
from Zulassungen
group by Marke
 - ◆ **select** Bundesland, sum(Anzahl)
from Zulassungen
group by Bundesland

group by und having (3)

- Beispiele von eben (Forts.):
 - ◆ **select** Marke, Bundesland, sum(Anzahl)
from Zulassungen
group by Marke, Bundesland

group by und having (4)

- Ist diese Anfrage zulässig?
select Marke, Datum, sum(Anzahl)
from Zulassungen
group by Marke, Bundesland

group by und having (5)

- **having** ist Selektionsbedingung auf gruppierter Relation,
- darf Bezug nehmen auf
 - ◆ Gruppierungsattribute,
 - ◆ beliebige Aggregatfunktionen über Nicht-Gruppierungsattributen.
- Beispiel:
select Bundesland, count(*)
from Zulassungen
group by Bundesland
having sum(Anzahl) > 100

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

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
```

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

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 |

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

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 |

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

Gruppierung: Schritt 3

- select A, sum(D)

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




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

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

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 |

Gruppierung: Beispiele (1)

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

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

• Ergebnis:

| Anzahl | PANr |
|--------|------|
| 2 | 7754 |
| 1 | 4711 |
| 1 | 5588 |
| 2 | 9912 |

- Zugrundeliegende Relation:

| Ausleihe | PANr | BuchNr | ... |
|----------|------|--------|-----|
| | | | |

Was bedeutet dieselbe Anfrage ohne group-by Klausel?

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

Gruppierung: Beispiele (2)

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

• Ergebnis:

| | PANr |
|---|------|
| 2 | 7754 |
| 2 | 9912 |

- **select** Matrikelnummer **from** Prüft **group by** Matrikelnummer **having avg**(Note) < (**select avg**(Note) **from** Prüft)

- Zugrundeliegende Relation:

| Prüft | Matrikelnummer | Note | ... |
|-------|----------------|------|-----|
| | | | |

Quantoren und Mengenvergleiche (1)

Einleitung
 SQL-Kern
 Weitere Konstrukte
 - Übersicht
 - Operationen
 - Aggregation
 - Gruppierung
 - Quantoren/Selbstverbund
 - order-by
 - Nullwerte
 Änderungen
 Schluß

- Syntax

```
attribut θ{
  all | any | some } (select attribut
  from ... where ...)
```

- Bedeutung: **all** Allquantor, **any, some** Existenzquantoren, äquivalent.

- Beispiele:

```
select PANr, Immatrikulationsdatum
from Studenten
where Matrikelnummer = any (
  select Matrikelnummer from Prüft)
```

- Äquivalente Formulierung mit **in**?

- Anfrage mit **any**, die nicht ohne weiteres mit **in** formulierbar?

- Einleitung
- SQL-Kern
- Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
- Änderungen
- Schluß

Quantoren und Mengenvergleiche (2)

- **select** Note **from** Prüft
where Matrikelnummer = 'HRO-912291'
and Note \geq **all** (
select Note **from** Prüft
where Matrikelnummer = 'HRO-912291')
- Wie sieht äquivalente Anfrage aus, die ohne Allquantor und Schachtelung auskommt?

- Einleitung
- SQL-Kern
- Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
- Änderungen
- Schluß

Quantoren und Mengenvergleiche (3)

- **select** Note **from** Prüft
where Matrikelnummer = 'HRO-912291'
and Note \geq **all** (
select Note **from** Prüft
where Matrikelnummer = 'HRO-912291')
- Anwendbarkeit eingeschränkt:
Test auf Mengen-Gleichheit geht nicht.
Definition von 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.

| | | | | | |
|------------|--|------|--|--------|--|
| Buch_Autor | | ISBN | | Author | |
| | | | | | |

- Einleitung
- SQL-Kern
- Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
- Änderungen
- Schluß

Selbst-Verbund (1)

- 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.Author | BA_2.ISBN | BA_2.Author |
|---------------|-------------|---------------|-------------|
| 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 |

- Einleitung
- SQL-Kern
- Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
- Änderungen
- Schluß

Selbst-Verbund (2)

- „Alle Prüfer, die zwei oder mehr Studenten geprüft haben.“
- Zählen von Wertemengen

| | | | | | | | |
|-------|--|------|--|----------------|--|-----|--|
| Prüft | | PANr | | Matrikelnummer | | ... | |
| | | | | | | | |

```
select distinct X.PANr
from Prüft X, Prüft Y
where X.PANr = Y.PANr
and X.Matrikelnummer <>
Y.Matrikelnummer
```

- Wie die Anfrage formulieren bei mehr als zwei Studenten?

Einleitung
SQL-Kern
Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
Änderungen
Schluß

order-by Klausel (1)

- Menge von Tupeln ~> Liste.
- Syntax
`order by` attributliste
- Beispiel

```
select Matrikelnummer, Note  
from Prüft  
where V_Bezeichnung = 'Datenbanken I'  
order by Note asc
```
- Aufsteigend (**asc**)
oder absteigend (**desc**) sortieren.

Einleitung
SQL-Kern
Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
Änderungen
Schluß

order-by Klausel (2)

- Sortierung wird auf das Ergebnis der jeweils vorangehenden SFW-Anfrage angewendet, also FALSCH:

```
select Matrikelnummer  
from Prüft  
where V_Bezeichnung = 'Datenbanken I'  
order by Note (falsch!)
```
- Note kommt nicht in select-Klausel vor.

Einleitung
SQL-Kern
Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
Änderungen
Schluß

Behandlung von Nullwerten (1)

- 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**.

Einleitung
SQL-Kern
Weitere Konstrukte
- Übersicht
- Operationen
- Aggregation
- Gruppierung
- Quantoren/Selbstverbund
- order-by
- Nullwerte
Änderungen
Schluß

Behandlung von Nullwerten (2)

- Beispiel:
Stadt

| Name | Bundesland |
|-----------|------------|
| Frankfurt | Hessen |
| München | Bayern |
| Zürich | NULL |
- **select** Name **from** Stadt
where Bundesland == 'Hessen'
- **select** Name **from** Stadt
where not (Bundesland == 'Hessen')

Behandlung von Nullwerten (3)

- Selbst ein Vergleich $A=A$ ist bei Vorliegen von Nullwerten keine Tautologie mehr, sondern ergibt unknown. Derartiger Vergleich in where-Klausel also nicht eliminierbar.
- `select * from Person where Nachname=Nachname`
 nicht dasselbe wie
`select * from Person`

| Vorname | Nachname |
|---------|----------|
| Gunter | Saake |
| Klemens | NULL |

- Boolesche Ausdrücke basieren dann auf dreiwertiger Logik.

Behandlung von Nullwerten (4)

| <i>and</i> | true | unknown | false |
|------------|---------|---------|-------|
| true | true | unknown | false |
| unknown | unknown | unknown | false |
| false | false | false | false |

| <i>or</i> | true | unknown | false |
|-----------|------|---------|---------|
| true | true | true | true |
| unknown | true | unknown | unknown |
| false | true | unknown | false |

| <i>not</i> | |
|------------|---------|
| true | false |
| unknown | unknown |
| false | true |

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 neuen Ausleihung),
 - ◆ Mehrtupel-Operationen („Erhöhe das Gehalt aller Mitarbeiter um 4.5%.“)
- Ändern von Sichten – grundsätzlich – in gleicher Weise möglich.

update (1)

- 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+1000 where Gehalt < 5000`

update (2)

| Angestellte | Name | Gehalt |
|-------------|--------|--------|
| | Meyer | 4000 |
| | Bond | 7200 |
| | Schulz | 5400 |

- **update** Angestellte **set** Gehalt = 6000
where Name = 'Bond'
- Was leistet das folgende Statement?
update Angestellte **set** Gehalt = 3000

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
Nicht dasselbe wie
drop table Ausleihe

Laufendes Beispiel

| Ausleih | Invnr | Name |
|---------|-------|--------|
| | 4711 | Meyer |
| | 1201 | Schulz |
| | 0007 | Müller |
| | 4712 | Meyer |

| Bücher | Invnr | Titel | ISBN | Autor |
|--------|-------|--------------|-------|------------|
| | 0007 | Dr. No | 3-125 | James Bond |
| | 1201 | Objektbanken | 3-111 | Heuer |
| | 4711 | Datenbanken | 3-765 | Vossen |
| | 4712 | Datenbanken | 3-891 | Ullman |
| | 4717 | Pascal | 3-999 | Wirth |

insert (1)

- **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', 'Karajan')

insert (2)

- **insert into** basisrelation
[(attribut_1, ..., attribut_n)]
SQL-anfrage
- **insert into** Kunde (**select** LName, LAdr, 0
from Lieferant)

| | | | |
|---------|------|-----|-----------------|
| Ausleih | Name | Adr | AnzahlAuftraege |
| | ... | ... | ... |

| | | | |
|-----------|-------|------|---------|
| Lieferant | LName | LAdr | Produkt |
| | ... | ... | ... |

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 und IEF),
 - ◆ (ANSI/ISO) SQL2, als SQL-92 verabschiedet,
 - ◆ (ANSI/ISO) SQL3, als SQL:99 verabschiedet,
 - ◆ (ANSI/ISO) SQL4 (geplant).

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.

SQL-92 (1)

- 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**.

SQL-92 (2)

- 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.

Mögliche Prüfungsfragen

- Formulieren diverser Anfragen, auch komplexerer Natur, in SQL.
- Was ist eine Umbenennung im SQL-Kontext? Wann wird sie gebraucht?
- Geben Sie ein sinnvolles Beispiel für eine Anfrage, die eine having-Klausel enthält.
- Geben Sie ein Beispiel für eine Anfrage mit einer having-Klausel, bei der
 - ◆ man die having-Klausel durch eine where-Klausel ersetzen kann,
 - ◆ dies nicht so ist.
- Erläutern Sie, warum im SQL-Kontext ‚A == A‘ keine Tautologie ist.