we focus on students

# SQL-Anfragen

## Deklarative Anfragen

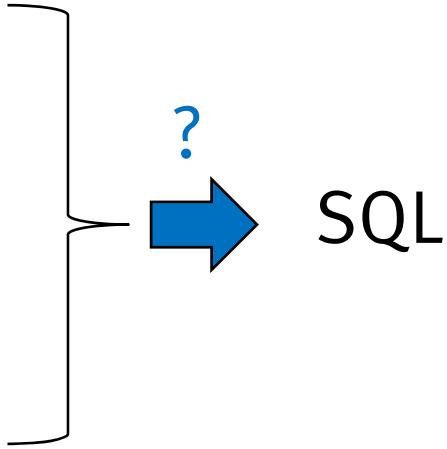
Fachhochschule Dortmund
University of Applied Sciences

© 2020 - Prof. Dr. Inga Marina Saatz

# Fragestellung

Relationale Algebra = Menge von Operatoren zur Manipulation von Relationen.

- Projektion
- Umbenennung
- Selektion
- Mengenoperatoren
- Verbundoperatoren



# Fragestellung



Deklarative Anfrage:
Was?

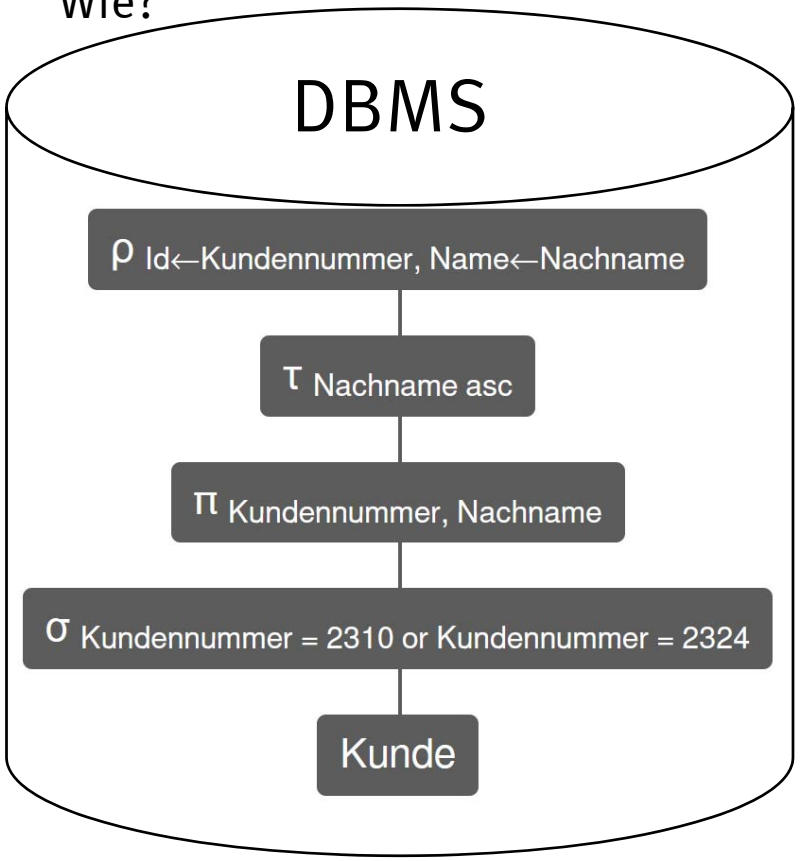
Kunde

| Kunden-nummer | Nach-name | Vorname |
|---|---|---|
| 2310 | Meitner | Anna |
| 2324 | Meier | Konstanze |
| 2343 | Schmidt | NULL |

Ausgabe

| ID | Name |
|---|---|
| 2324 | Meier |
| 2310 | Meitner |

Anfrageoptimierung:
Wie?



# Beschreibung von Anfragen



| | Relationale Operatoren |
|---|---|
| Was wird gesucht? | Projektion (Auswahl der Attribute) |
| In welchen Relationen? | Angabe der Relation(en) |
| Auswahlbedingungen? | Selektion (Auswahl der Tupel) |
| Sortierung? | Sortierreihenfolge der Tupel |
| Geänderte Bezeichnungen? | Umbenennung |

# Beschreibung von Anfragen

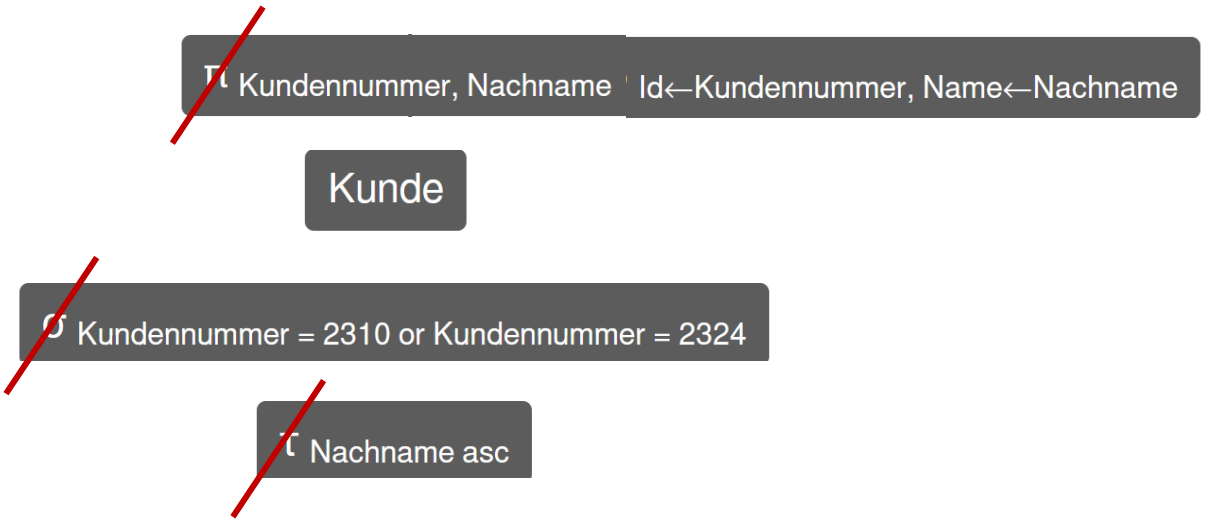


**SELECT**

**FROM**

**WHERE**

**ORDER BY**



# Beschreibung von Anfragen



**SELECT** SELECT **k.**Kundennummer AS Id, **k.**Nachname AS Name

**FROM** FROM Kunde **k** Tabellen-ALIAS

**WHERE** WHERE **k.**Kundennummer=2310 OR **k.**Kundennumer=2324

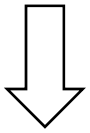
**ORDER BY** ORDER BY **k.**Nachname ASC

# Alternativen



Kunde

| Kunden-nummer | Nach-name | Vorname |
|---|---|---|
| 2310 | Meitner | Anna |
| 2324 | Meier | Konstanze |
| 2343 | Schmidt | NULL |



Ausgabe

| ID | Name |
|---|---|
| 2324 | Meier |
| 2310 | Meitner |

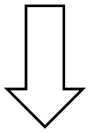
```
SELECT Kundennummer AS Id, Nachname AS Name
FROM   Kunde
WHERE Kundennummer=2310 OR Kundennumer=2324
ORDER BY Nachname ASC
```

# Alternativen



Kunde

| Kunden-nummer | Nach-name | Vorname |
|---|---|---|
| 2310 | Meitner | Anna |
| 2324 | Meier | Konstanze |
| 2343 | Schmidt | NULL |



Ausgabe

| ID | Name |
|---|---|
| 2324 | Meier |
| 2310 | Meitner |

```
SELECT Kundennummer AS Id, Nachname AS Name
FROM   Kunde
WHERE Kundennummer IN (2310, 2324)
ORDER BY Nachname ASC
```

# Vergleich von Zeichenketten



- Zeichenkettenvergleich mit **LIKE**
  - % : kein oder beliebig viele Zeichen
  - _ : für genau ein Zeichen
- Beispiele
  - Welche Namen beginnen mit ‚Mei' und endet auf ‚er' ?

```
SELECT      Kundennummer, Nachname
FROM        Kunde
WHERE       Nachname LIKE  'Mei%er'
```

  - Mayer? Maier? Meier? Meyer? …?

```
SELECT      Kundennummer, Nachname
FROM        Kunde
WHERE       UPPER(Nachname) LIKE 'M_ _ER%'
```

*Konvertierung in Großbuchstaben*

*Nachname CHAR(30)*

# Reguläre Ausdrücke



Reguläre Ausdrücke können in der Where-Klausel verwendet werden.
Der Beginn des regulären Ausdrucks wird durch das Zeichen **^** gekennzeichnet.
Der reguläre Ausdruck wird mit dem Zeichen $ abgeschlossen.

- Beispiele
  - Die Kundennummer des Kunden besteht nur aus Zahlen:

```
SELECT Kundennummer, Nachname
FROM Kunde
WHERE REGEXP_LIKE(Kundennummer, '^[[:digit:]]+$')
```

  - Das Nachname des Kunden besteht nur aus Buchstaben:

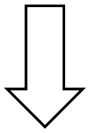
```
SELECT Kundennummer, Nachname
FROM Kunde
WHERE REGEXP_LIKE(Nachname, '^[A-Za-z]+$')
```

# Alternativen



Kunde

| Kunden-nummer | Nach-name | Vorname |
|---|---|---|
| 2310 | Meitner | Anna |
| 2324 | Meier | Konstanze |
| 2343 | Schmidt | NULL |



Ausgabe

| ID | Name |
|---|---|
| 2324 | Meier |
| 2310 | Meitner |

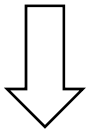
```
SELECT Kundennummer AS Id, Nachname AS Name
FROM   Kunde
WHERE Vorname IS NOT NULL
ORDER BY Nachname ASC
```

# Alternativen



Kunde

| Kunden-nummer | Nach-name | Vorname |
|---|---|---|
| 2310 | Meitner | Anna |
| 2324 | Meier | Konstanze |
| 2343 | Schmidt | NULL |



Ausgabe

| ID | Name |
|---|---|
| 2343 | Schmidt |

```
SELECT Kundennummer AS Id, Nachname AS Name
FROM   Kunde
WHERE Vorname IS ~~NOT~~ NULL
ORDER BY Nachname ASC
```

# Zusammenfassung



Grundform einer SQL-Abfrage

Was wird gesucht?

In welchen Tabellen?

Auswahlbedingungen?

Sortierung?

**SELECT [DISTINCT]** ‹Spalte$_1$›, ...,‹Spalte$_n$›

Projektion (Festlegung der Ausgabespalten)

**FROM** ‹Tabelle$_1$›, ..., ‹Tabelle$_m$›

Join (Angabe der Tabellen und Verbundbedingung)

**WHERE** ‹Bedingung›

Selektionsbedingung (Auswahl der Tupel)
– optional

**ORDER BY** ‹Spalte$_1$›, ...,‹Spalte$_n$› {ASC|DESC}

Sortierreihenfolge der Tupel in der Ergebnistabelle
– optional

we focus on students

# Datenbankanfragen

## Gruppierung

Fachhochschule Dortmund
University of Applied Sciences

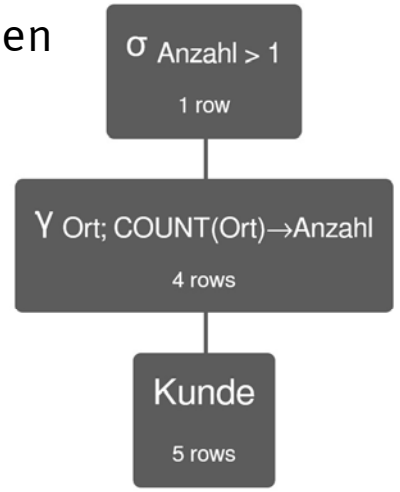
© 2020 - Prof. Dr. Inga Marina Saatz

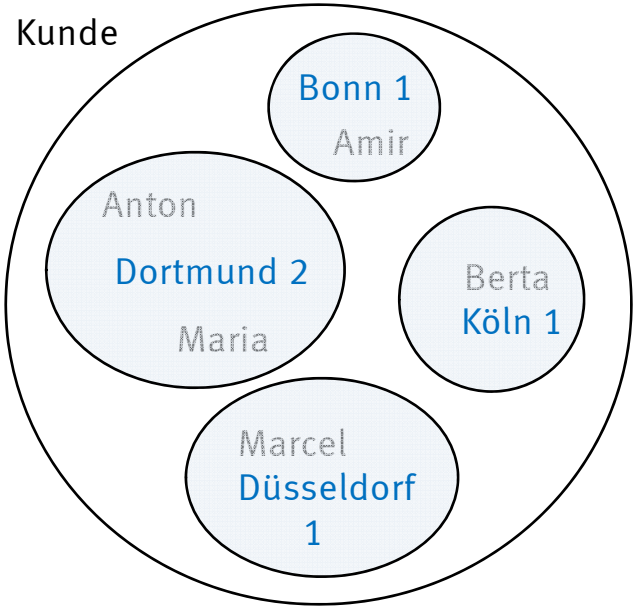
# Fragestellung



Zähle alle Kunden pro Orte mit mehr als einem Kunden

1. Gruppierung nach dem Ort
2. Zählen der Tupel pro Untergruppe
3. Selektiere die Orte mit mehr als einem Kunden




σ Anzahl > 1 γ Ort; COUNT(Ort)→Anzahl Kunde

| Ort | Anzahl |
|---|---|
| Dortmund | 2 |

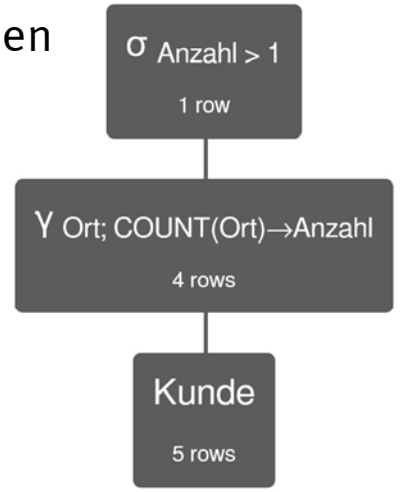
Wie lautet die SQL-Anfrage?

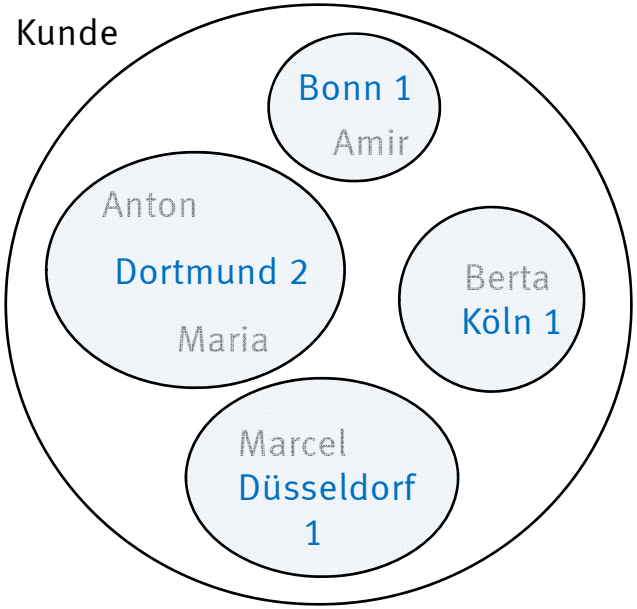
# Lösung



Zähle alle Kunden pro Orte mit mehr als einem Kunden

1. Gruppierung nach dem Ort
2. Zählen der Tupel pro Untergruppe
3. Selektiere die Orte mit mehr als einem Kunden




σ Anzahl > 1 γ Ort; COUNT(Ort)→Anzahl Kunde

| Ort | Anzahl |
|---|---|
| Dortmund | 2 |

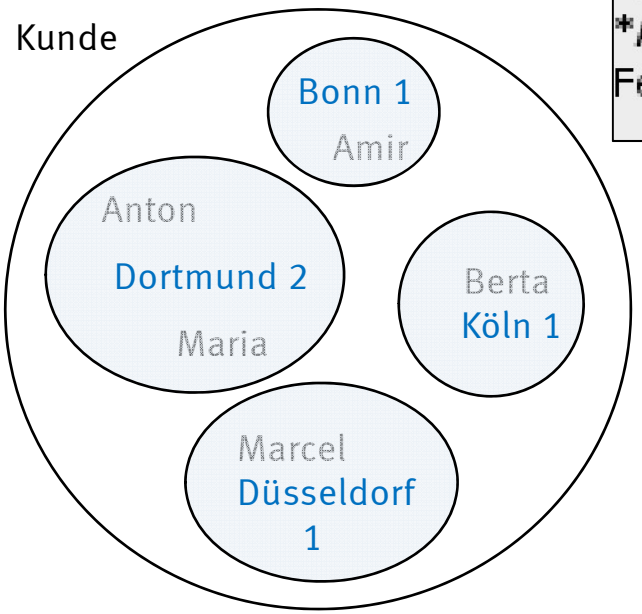
```
SELECT Ort, COUNT(*) AS Anzahl
FROM   Kunde
GROUP BY Ort
HAVING COUNT(*) › 1
```

# WHERE-Klausel vs. Having-Klausel



Zähle alle Geschäftskunden (Anrede IS NULL)
in den Orten mit mehr als einem Geschäftskunden

```
SELECT Ort, COUNT(*) AS Anzahl
FROM   Kunde
WHERE COUNT(*) > 1 AND Anrede IS NULL
GROUP BY Ort
```




```
ORA-00934: Gruppenfunktion ist hier nicht zulässig
00934. 00000 -  "group function is not allowed here"
*Cause:
*Action:
Fehler in Zeile: 3 Spalte: 7
```

Zuerst Gruppieren,
dann zählen!

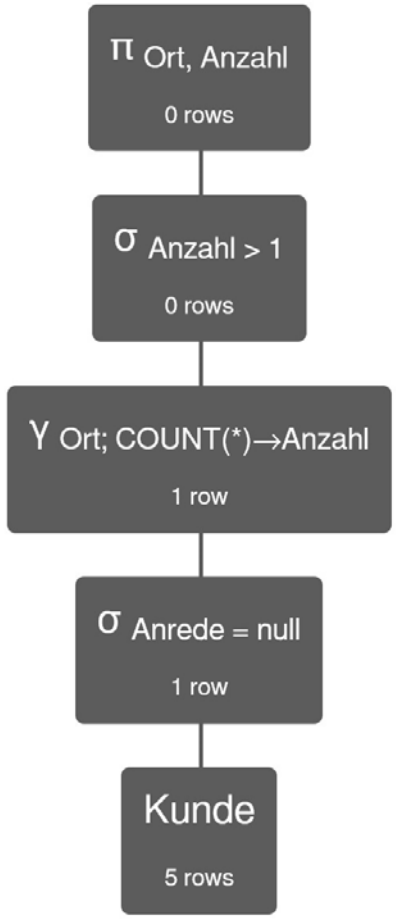
# WHERE-Klausel vs. Having-Klausel



Zähle alle Geschäftskunden (Anrede IS NULL)
in den Orten mit mehr als einem Geschäftskunden

| Ort | Anzahl |
|---|---|

| Kunden-nummer | Anrede | Vorname | Nach-name |
|---|---|---|---|




HAVING

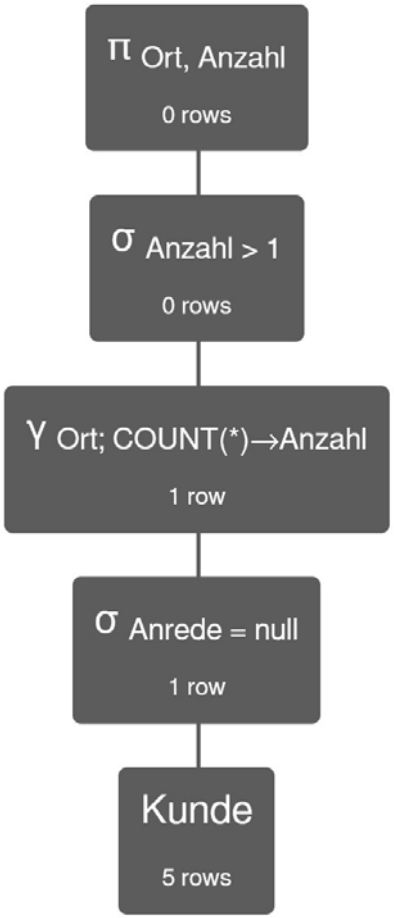
GROUP BY

WHERE

# WHERE-Klausel vs. Having-Klausel



Zähle alle Geschäftskunden (Anrede IS NULL)
in den Orten mit mehr als einem Geschäftskunden




HAVING

GROUP BY

WHERE

```
SELECT Ort, COUNT(*) AS Anzahl
FROM   Kunde
WHERE Anrede IS NULL
GROUP BY Ort
HAVING COUNT(*) › 1
```

# Aggregationsfunktionen



- Arithmetischer Mittelwert (Durchschnitt) **AVG(.)**
- Anzahl der Zeilen **COUNT(.)**
- Maximum (auch alphanum.) **MAX(.)**
- Minimum (auch alphanum.) **MIN(.)**
- Summenbildung **SUM(.)**

Verwendung:

```
SELECT COUNT(Kundennummer)
FROM   Kunde
WHERE Nachname LIKE 'Mei%'
```

```
SELECT COUNT(Anrede)
FROM   Kunde
WHERE Nachname LIKE 'Mei%'
```

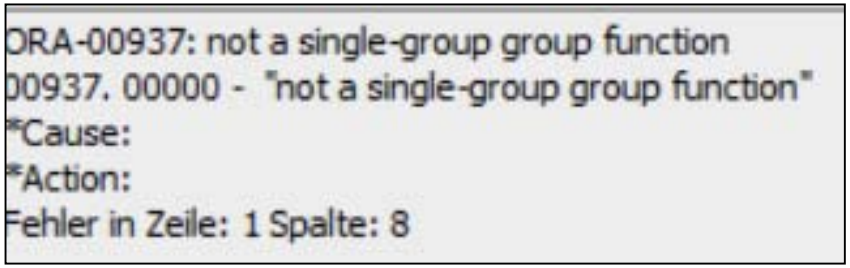
Tupel mit Anrede Null werden nicht mitgezählt!

# Typische Fehler



## Gruppierung vergessen

```
SELECT Ort, COUNT(*) AS Anzahl
FROM   Kunde
GROUP BY Ort
```

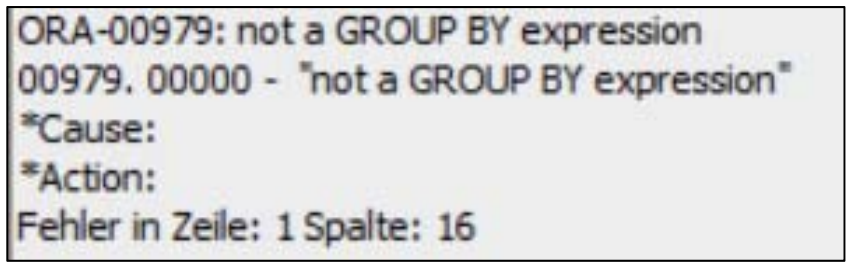



Single Value Rule (SQL-Standard SQL92)
Alle nicht-berechneten Attribute **müssen** in die Group-By-Klausel aufgenommen werden.

# Typische Fehler

## Zugriff auf ein nicht gruppiertes Attribut

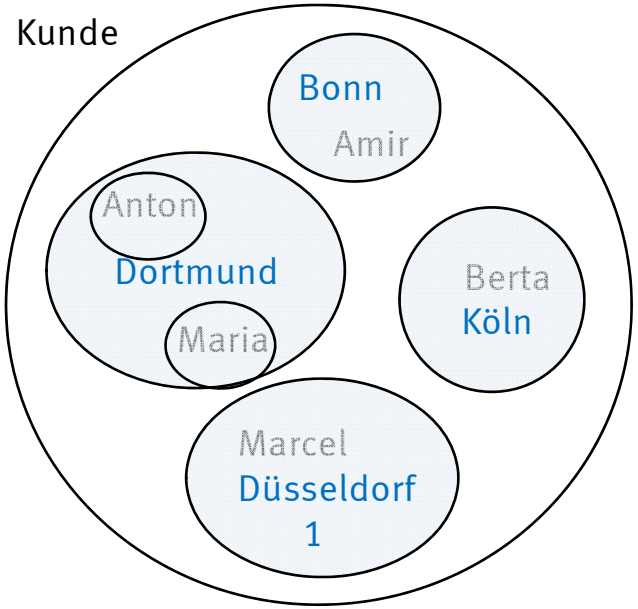
```
SELECT Ort, COUNT(*) AS Anzahl
FROM   Kunde
GROUP BY Ort
ORDER BY Ort, Vorname ASC
```



# Typische Fehler

## Fehlende Attribute in der Gruppierung

```
SELECT Ort, Vorname,  COUNT(*) AS Anzahl
FROM    Kunde
GROUP BY Ort, Vorname
ORDER BY Ort, Vorname ASC
```




| Ort | Vorname | Anzahl |
|---|---|---|
| Bonn | Amir | 1 |
| Dortmund | Anton | 1 |
| Dortmund | Maria | 1 |
| Düsseldorf | Marcel | 1 |
| Köln | Berta | 1 |

# Syntax der SELECT-Anweisung



Was wird gesucht?

**SELECT** ‹Spalte$_1$›, ..., ‹Spalte$_n$›

Projektion (Festlegung der Ausgabespalten)

In welchen Tabellen?

**FROM** ‹Tabelle$_1$›, ..., **‹Tabelle$_m$›**

Join (Angabe der Tabellen und Verbundbedingung)

Auswahlbedingungen?

**WHERE** ‹Bedingung›

Selektionsbedingung (Auswahl der Tupel)
– optional

Gruppierung erforderlich?

**GROUP BY** ‹Spalte$_1$›, ..., ‹Spalte$_n$›

Gruppenbildung mit gleichen Werten
– optional

Gruppierungsbedingung?

**HAVING** ‹Bedingung›

Selektion von Gruppen
– optional unter group by

Sortierung?

**ORDER BY** ‹Attribut-Liste›

Sortierreihenfolge der Tupel in der Ergebnistabelle
– optional

# Datenbankanfragen

## Verbundoperationen

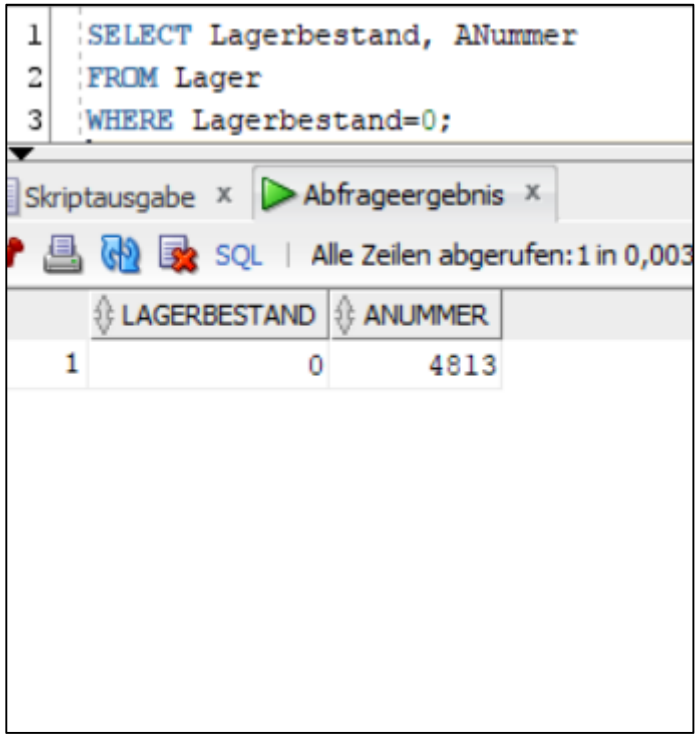
Fachhochschule
Dortmund
University of Applied Sciences

© 2020 - Prof. Dr. Inga Marina Saatz

# Fragestellung



**SELECT** ‹Spalte1›, ..., ‹SpalteN›
FROM ‹Tabelle1› [, ..., ‹TabelleM›]
WHERE ‹Selektionsbedingung›




Wie kann eine SQL-Abfrage über mehrere Tabellen formuliert werden?

# Kreuzprodukt

**SELECT** ‹Spalte1›, ..., ‹SpalteN›
FROM ‹Tabelle1› [, ..., ‹TabelleM›]
WHERE ‹Selektionsbedingung›

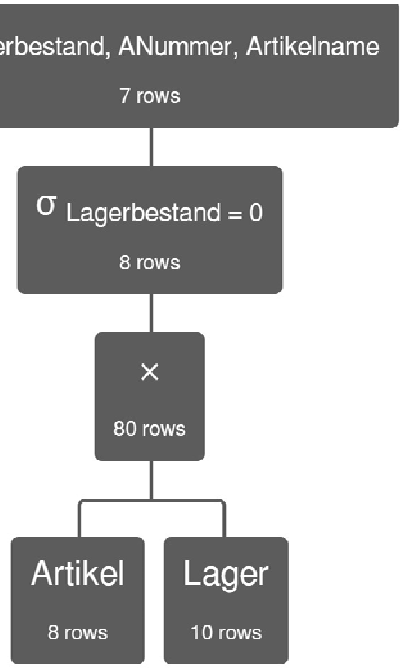
zu viele Tupel !!!

```
1  SELECT Lagerbestand, ANummer, Artikelname
2  FROM Lager, Artikel
3  WHERE Lagerbestand=0;
```

Skriptausgabe | Abfrageergebnis | SQL | Alle Zeilen abgerufen:8 in 0,004 Sekunden

| | LAGERBESTAND | ANUMMER | ARTIKELNAME |
|---|---|---|---|
| 1 | 0 | 4813 | Datenbanksysteme |
| 2 | 0 | 4813 | Datenbanksysteme |
| 3 | 0 | 4813 | Märchen von Beedle dem Barden |
| 4 | 0 | 4813 | Anatomie-interaktiv |
| 5 | 0 | 4813 | Anatomie |
| 6 | 0 | 4813 | Anatomie-Atlas |
| 7 | 0 | 4813 | Datenbank-Skript |
| 8 | 0 | 4813 | Harry Potter Band 20 |

Kreuzprodukt



# INNER-JOIN




```
SELECT Artikelname, Autor, Ausgabe
FROM    Artikel a, Lager l
WHERE Lagerbestand=0
AND a.Artikelnummer = l.ANummer
```

| Artikel-nummer | Artikel name | Preis | Ausgabe | Lager-nummer | Standort | ANummer | Lager-bestand |
|---|---|---|---|---|---|---|---|
| 4812 | Basisw | 19,95 | broschiert | 27135 | R235 | 4812 | 0 |
| 4813 | Das End | 10,00 | broschiert | 27135 | R235 | 4812 | 0 |

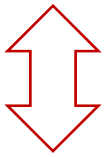
Menge A

Menge L

# Beispiel INNER-JOIN




```
SELECT Artikelname, Autor, Ausgabe
FROM    Artikel a, Lager l
WHERE Lagerbestand=0
AND a.Artikelnummer = l.ANummer
```

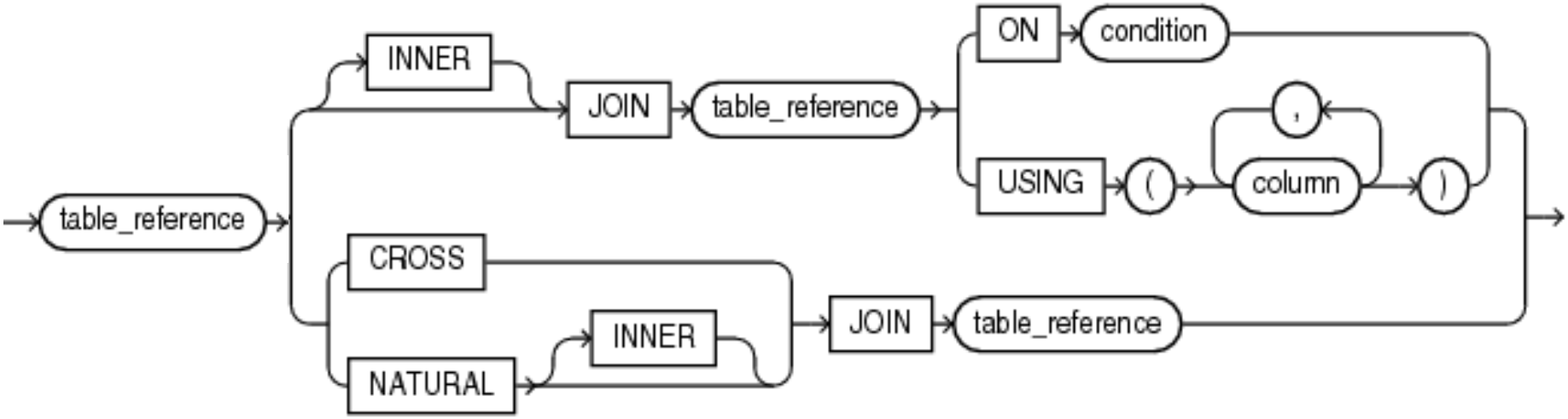


*semantisch äquivalent*

```
SELECT Artikelname, Autor, Ausgabe
FROM    Artikel a JOIN Lager l
      ON a.Artikelnummer = l.ANummer
WHERE Lagerbestand=0
```

# Syntaxbeschreibung: Inner Join

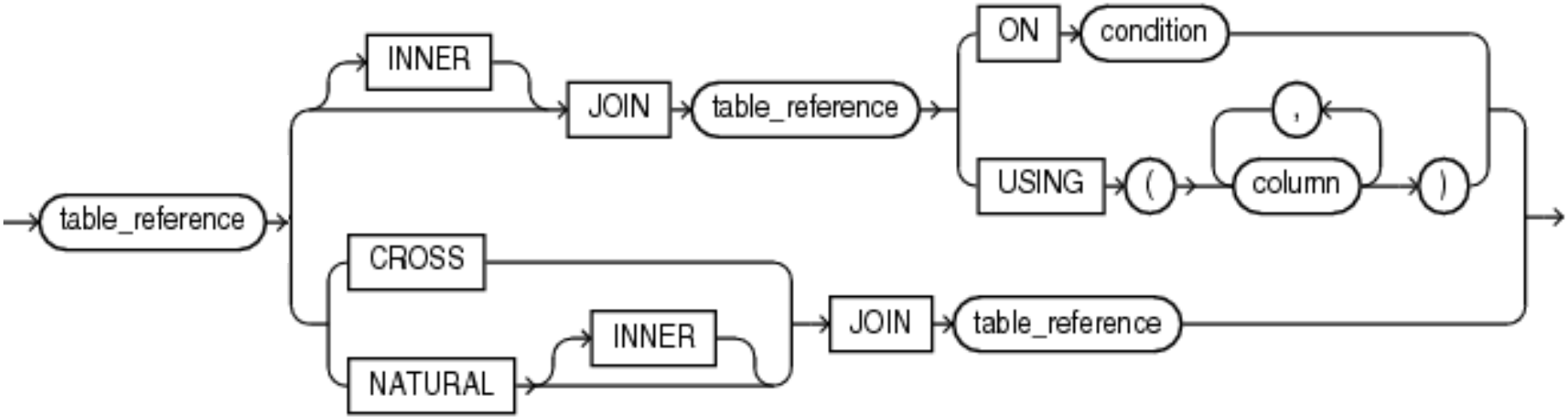





```
SELECT Artikelname, Autor, Ausgabe
FROM    Artikel a JOIN Lager l
        ON a.Artikelnummer = l.ANummer
WHERE Lagerbestand=0
```

# Syntaxbeschreibung: Inner Join



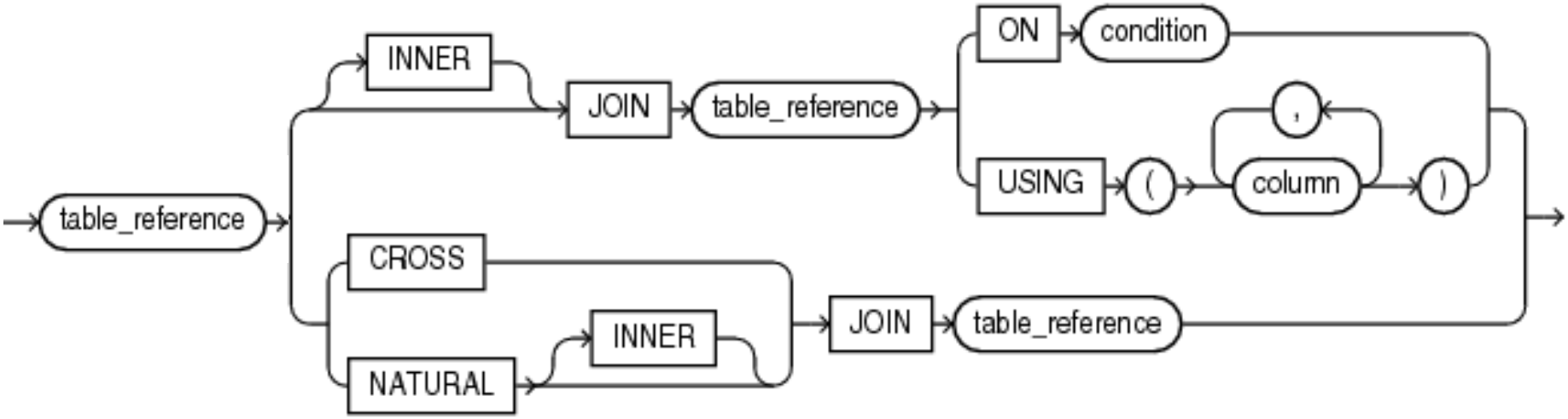



```
SELECT w.Artikelnummer, Artikelname, Autor
FROM   Artikel a JOIN Warenkorb w
       USING (Artikelnummer)
```

Attributliste möglich

# Syntaxbeschreibung: Inner Join



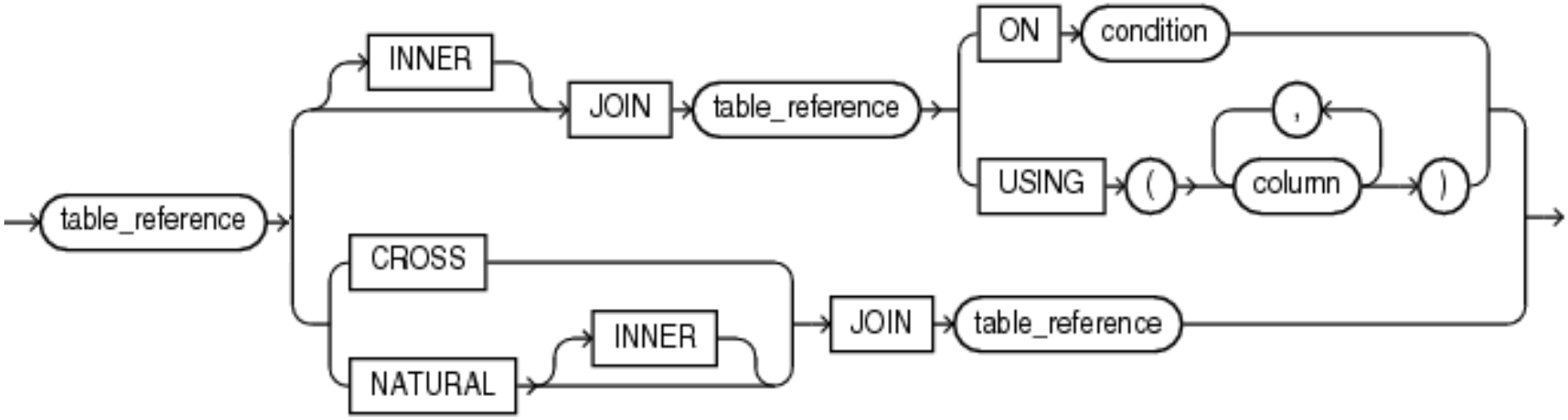



```
SELECT w.Artikelnummer, Artikelname, Autor
FROM   Artikel a NATURAL JOIN Warenkorb w
```

Verbundbedingung:
Gleichheit der gleichbenannten Attribute

# Verbund über mehrere Tabellen

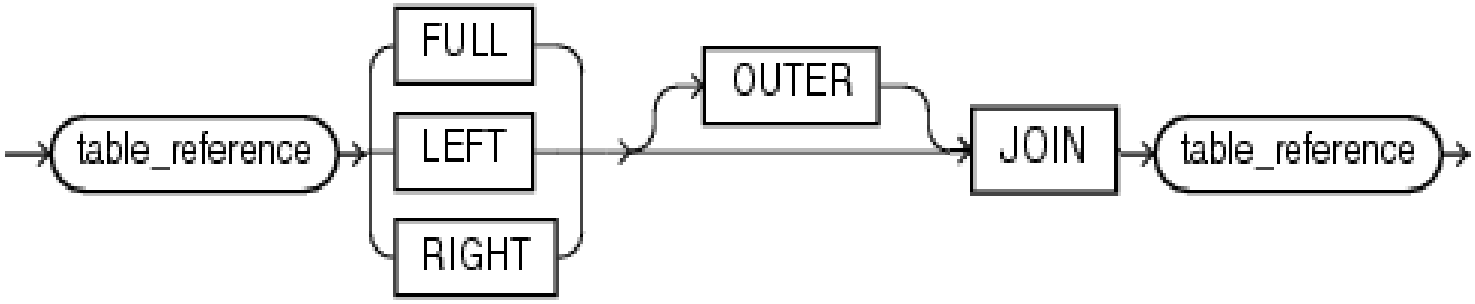





```
SELECT Nachname, Artikelname, Anzahl
FROM   Kunde JOIN Warenkorb USING (Kundennummer)
       NATURAL JOIN Artikel
WHERE Kundennummer=2310
```

# Outer Join



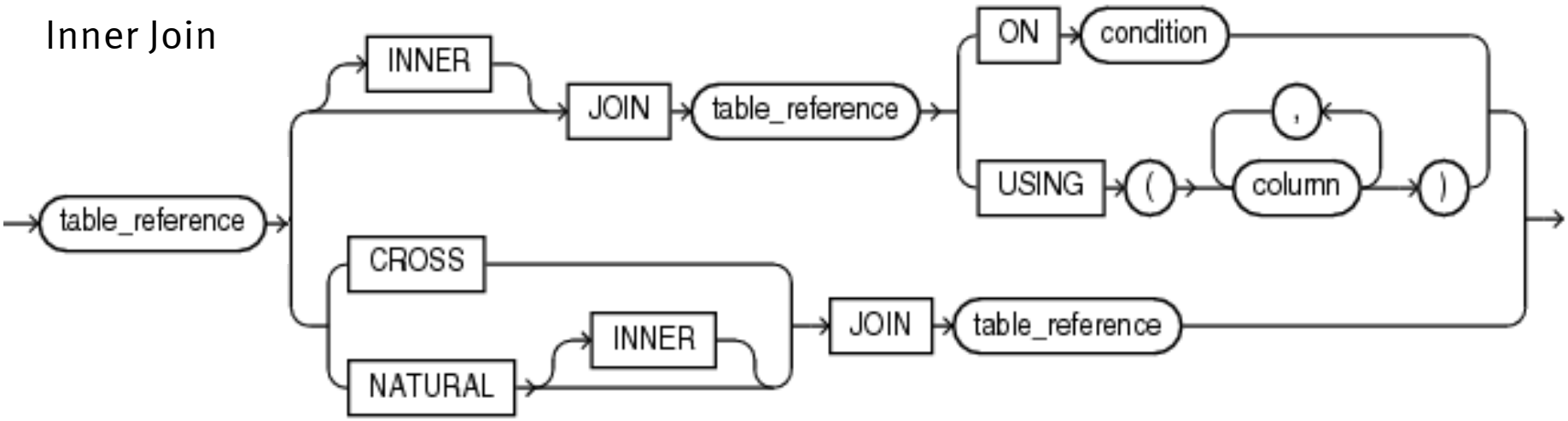



```
SELECT Artikelname, Autor, Ausgabe
FROM   Artikel a LEFT JOIN Lager l
       ON a.Artikelnummer = l.ANummer
WHERE Lagernummer IS NULL
```

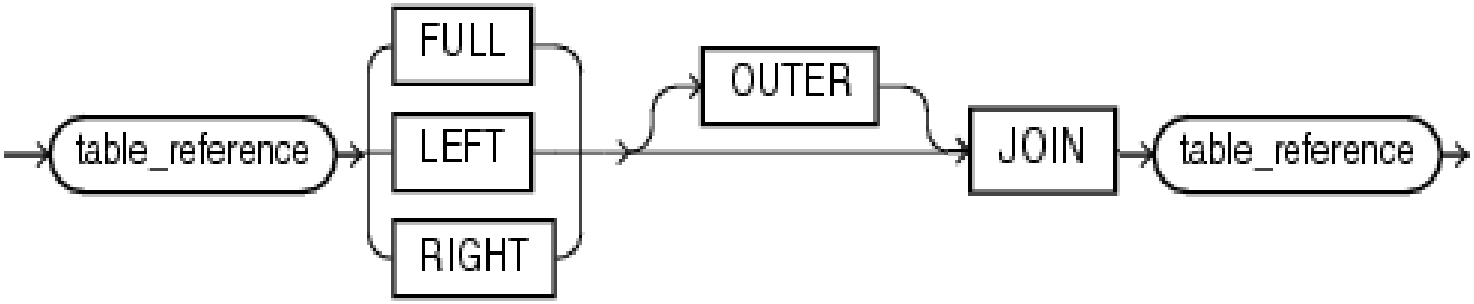
# Zusammenfassung



Inner Join




Outer Join



# Alternative Syntaxbeschreibung



*joined_table*: {
*table_reference* {[INNER | CROSS] JOIN} *table_factor* [*join_specification*] |
*table_reference* {LEFT|RIGHT} [OUTER] JOIN *table_reference join_specification* |
*table_reference* NATURAL [INNER | {LEFT|RIGHT} [OUTER]] JOIN *table_factor*
}

*join_specification*: {
ON *search_condition* |
USING (*join_column_list*)
}

*join_column_list*:
*column_name* [, *column_name*] ...

we
focus
on
students

# Datenbankanfragen

## Funktionen nutzen

Fachhochschule
Dortmund
University of Applied Sciences

© 2020 - Prof. Dr. Inga Marina Saatz

# Zusammenfassung

Beispiel:

```
select Round(122.716, -1)
from dual;
```

| ROUND(122.716,-1) |
|---|
| 120 |

-1 bedeutet, dass auf die 1. Stelle vor dem Komma gerundet wird.

Die Hilfstabelle dual des DBMS Oracle besitzt ein Attribut und genau ein Tupel.

dual

| | DUMMY |
|---|---|
| 1 | X |