

---

# **Datenbanken 1: Datenbankanfragen (Teil 1)**

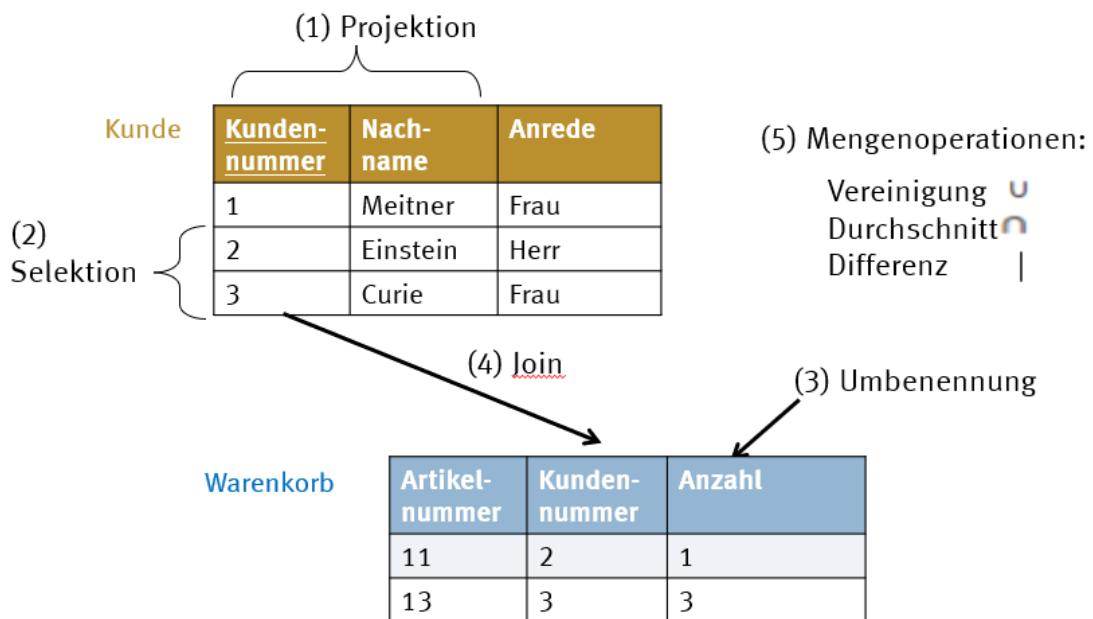
**Prof. Dr. Inga Saatz**

**Oct 29, 2023**

**Note:** Nach dieser Vorlesung sollten Sie:

- Anfragen der relationalen Algebra in SQL-Anfragen übersetzen können.
- Die Grundstruktur und Syntax von SQL-Anfragen kennen.
- Einfache SQL-Anfragen formulieren können.
- Herstellerspezifische Funktionen bei der Formulierung von Auswahlbedingungen verwenden können.

## Übersetzung der relationalen Operatoren nach SQL



Grundform einer SQL-Anfrage

```

SELECT <Spalte1>, ..., <SpalteN>           Projektion:      Was wird gesucht?
FROM   <Tabelle1>, ..., <Tabellej>          Tabellen:        Wo wird gesucht?
WHERE  <Bedingung>                          Selektion:      Welche Tupel werden
  gesucht?
ORDER BY <SpalteK> {ASC | DESC}, ...        Sortierung:    Wie sollen die Tupel
  sortiert werden?

```

## Von der relationalen Algebra zu SQL

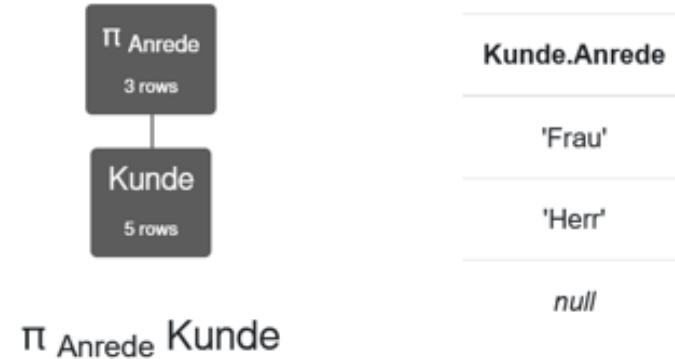
```

# Laden der iSQL-Extension und Herstellung der Datenbankverbindung in Python
%load_ext sql
%sql oracle+oracledb://C##FBPOOL2:oracle@172.22.160.22:1521?service_name=xe

```

## 1. Projektion

In der relationalen Algebra werden **Duplikate** aus der Ergebnismenge immer entfernt.



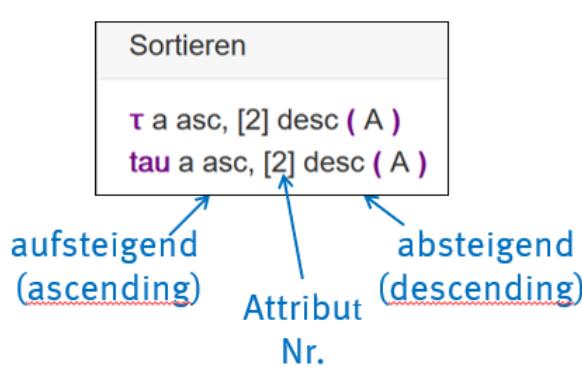
In SQL werden doppelte Tupel angezeigt, da sich die Tupel in ihrer Zeilennummer in der Datenbasis unterscheiden. Die Zeilennummer kann über das Pseudo-Attribut `ROWNUM` in der `SELECT`-Klausel angezeigt werden.

In SQL werden mit dem Schlüsselwort `DISTINCT` doppelte Tupel entfernt.

```
SELECT [DISTINCT] <Spalte1>, ..., <SpalteN>
FROM   <Tabelle1> [, ..., <TabelleM>]
```

### Sortierung

In der relationalen Algebra:



In SQL:

```
SELECT <Spalte1>, ..., <SpalteN>
FROM <Tabelle1> [, ..., <TabelleM>]
ORDER BY <SpalteK> {ASC | DESC}, ...
```

**aufsteigend  
(ascending)**      **absteigend  
(descending)**

Beispiel:

Liste alle Kunden mit Kundennummer, Anrede und Nachname absteigend sortiert nach dem Nachnamen

### Beispiel

Liste alle Kunden mit Kundennummer, Anrede und Nachname absteigend sortiert nach dem Nachnamen.

## 2. Selektion

- In Auswahlbedingungen können Vergleichsoperatoren verwendet werden (<, >, =, !=, <=, >=).
- Die Verknüpfung von Auswahlbedingungen erfolgt durch logische Operatoren (AND, OR, NOT).

### Beispiel

Liste alle weiblichen Kunden aus Dortmund.

NULL-Werte Die Prüfung auf einen NULL-Wert erfolgt mit dem IS Operator:

<attribut> IS NULL

**Warning:** Weshalb können NULL-Werte nicht auf Gleichheit geprüft werden? Ein NULL-Wert beschreibt die Situation, dass ein Wert nicht vorhanden oder zutreffend ist. Daher können NULL-Werte nicht mit dem Gleich-Operator “=” mit einem Wert verglichen werden.

**Note:** Python vs. Java In Python wird der NULL-Wert durch den Wert `None` dargestellt.

### Beispiel

Finde alle Geschäftskunden. Überlegen Sie sich zunächst, was einen Geschäftskunden auszeichnet.

### Beispiel

Finde alle Kunden, die *keine* Geschäftskunden sind.

## 3. Umbenennung von Spalten

In der relationalen Algebra:

Umbenennung des Attributs a der Tabelle A in y mit dem  $\rho$ -Operator:

Umbenennung von Spalten

$\sigma A.y > 2 (\rho y \leftarrow a (A))$

In SQL werden zur Umbenennung Alias verwendet.

Alias Ein Alias ist eine alternative Bezeichnung für ein Attribut oder eine Tabelle in einer SQL-Anfrage. Ein Alias wird mit dem Schlüsselwort AS vergeben. In der FROM- und der SELECT-Klausel ist das Schlüsselwort nicht erforderlich.

Spalten-Alias  
SELECT w.Kundennummer, Anzahl **AS Artikelanzahl**  
FROM Warenkorb w ORDER BY Artikelanzahl DESC  
Tabellen-Alias

### Beispiel

Geben Sie alle männlichen Kunden aus mit Kundennummer und Nachname.

Bei der Ausgabe soll das Attribut Nachname in Name umbenannt und es soll zudem ein Tabellen-Alias verwendet werden.

## 4. Abfragen auf mehreren Tabellen

**Warning:** Achtung! Verbundbedingungen sind bei Abfragen über mehrere Tabellen sind notwendig, da durch die Auflistung der Tabellen in der FROM-Klausel das kartesische Produkt der Tabellen gebildet wird.

```
SELECT <Spalte1>, ..., <SpalteN>
FROM   <Tabelle1>, ..., <Tabellej>      -- Kartesische Produkt der Tabelle
WHERE  <Bedingung>                      -- Verbundbedingungen
ORDER BY <SpalteK> {ASC | DESC}, ...
```

**Note:** Alternative Formulierungen Die Verwendung von Join-Operatoren ist auch möglich. Hierauf wird in der Vorlesung SQL-Anfragen (Teil 2) genauer eingegangen.

### Beispiel

Welche Artikel haben die weiblichen Kunden im Warenkorb?

Bitte geben Sie die Nachnamen und die Artikelnamen aufsteigend sortiert nach dem Kundennamen an.

### Beispiel

Liste die Artikel mit Artikelbezeichnungen und Lagerbestand und gebe nur die Artikel aus, die auf Lager sind.

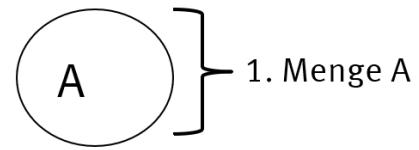
Die Ausgabe soll aufsteigend nach dem Lagerbestand und absteigend nach dem Artikelbezeichnung sortiert werden.

## 5. Mengenoperationen

Die Anwendung einer Mengenoperation in SQL ist auf gleichstrukturierten Ergebnismengen von SQL-Anfragen möglich.

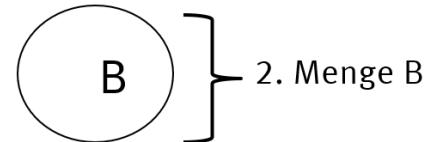
Mengenoperation	Schlüsselwort in SQL
Vereinigung	UNION (ohne Duplikate), UNION ALL (mit Duplikate)
Schnittmenge	INTERSECT
Differenzmenge	MINUS

```
SELECT  <Spalte1>, ..., <Spalten>
FROM    <Tabelle1>, ..., <Tabellem>
WHERE   <Bedingung- A>
```



{UNION | UNION ALL | INTERSECT | MINUS }

```
SELECT  <Spalte1>, ..., <Spalten>
FROM    <Tabelle1>, ..., <Tabellem>
WHERE   <Bedingung- B>
```



```
ORDER BY <Spalte1> ...
```

Sortierung

**Attention:** Sortierung Die Sortierung erfolgt erst **nach** Ausführung der Mengenoperation am Ende der SQL-Anfrage.

### Beispiel

Welche Kunden haben nichts in ihrem Warenkorb?  
Die Ausgabe soll aufsteigend nach den Nachnamen sortiert werden.

### Tipp

1. Wie lautet die SQL-Anfrage, die alle Kunden liefert?
2. Wie lautet die SQL-Anfrage, welche die Kunden liefert, die etwas bestellt haben?
3. Welcher Mengenoperator wird jetzt benötigt, um beide Anfragen zu kombinieren?

### Herstellerspezifische Funktionen

### Numerischen Funktionen

Funktion	Funktion	Beispiel
Kaufm. Runden	ROUND (zahl, anzahl_stellen)	ROUND(122.716, 2) = 122,72 ROUND(122.716, -2) = 100
Abrunden (auf Ganzzahl)	FLOOR (zahl)	FLOOR (122.716) = 122
Aufrunden	CEIL(zahl)	CEIL (122.716) = 123
Betrag	ABS(zahl)	ABS (-122.716) = 122.716
Vorzeichen	SIGN(zahl)	SIGN (-122.716) = -1
Modulo	MOD(zahl1, zahl2)	MOD (122, 80) = 42

### Beispiel

Runde die Zahl 122,57 kaufmännisch auf

- eine Stelle nach dem Komma
- ganze Zahlen
- auf Hunderter

**Hint:** Die Tabelle dual Für die Abfrage von Systemvariablen und für Berechnungen kann die standardmäßig in Oracle vorhandene Hilfstabelle dual genutzt werden. Sie besitzt ein Attribut und genau ein Tupel.

### Beispiel

Bestimme den größten und den kleinsten Wert aus der Liste (12, '7', 3, '15', 8).  
Finden Sie zunächst geeignete Oracle-Funktionen (z.B. in der Oracle-Dokumentation).

### Tipp ?

Betrachten Sie die Oracle-Funktionen MIN(), LEAST(), MAX() und GREATEST().  
Welche dieser Funktionen passen zu dieser Aufgabenstellung?

### Aggregationsfunktionen

Durch Aggregationsfunktionen berechnen zusammenfassende (=aggregierende) Werte über eine Spalte der Ergebnismenge, beispielsweise die Anzahl aller Elemente.

Aggregationsfunktion	Wirkung
AVG(.)	liefert den arithmetischer Mittelwert (Durchschnitt) über eine Spalte von allen Tupeln
COUNT(.)	liefert die Anzahl der Tupel (Zeilen)
MAX(.)	liefert das Maximum (auch alphanum.) über eine Spalte von allen Tupeln
MIN(.)	liefert das Minimum (auch alphanum.) über eine Spalte von allen Tupeln
SUM(.)	bildet die Summe über eine Spalte von allen Tupel

**Warning:** Achtung NULL-Werte werden nicht berücksichtigt!

### Beispiel

Ermittle die Anzahl aller weiblichen Kunden.

### Beispiel

Ermittle die Anzahl der Artikel im Warenkorb sowie den zugehörenden Warenwert zum Kunden mit der Kundennummer 7562.

## Zeichenketten-Funktionen

### Datentypen

CHAR, CHAR(n)

- Statischer Datentyp für Zeichenketten **konstanter** Länge.
- CHAR: Länge 1, CHAR(n) Länge n Bytes
- Maximale Größe 2000 Bytes
- Nutzung:
  - CHAR(n): Zeichenketten fester Länge
  - CHAR: Zeichenkette der Länge 1
- Vorteil:
  - 50% schnellerer Zugriff als bei VARCHAR2
- Nachteil:
  - Speicherplatzverschwendungen bei variabler Länge

VARCHAR(n), VARCHAR2(n)

- Dynamischer Datentyp für Zeichenketten **variabler** Länge (bis n Bytes)

- Die Länge n **muss** angegeben werden
- Maximale Größe 4000 Bytes
- Nutzung:
  - VARCHAR2 Zeichenketten variabler Länge
  - VARCHAR ist deprecated
- Vorteil:
  - Effiziente Speicherplatznutzung
- Nachteil:
  - Im Vergleich langsamer als CHAR(n)

**Hint:** Beispiel

```
CREATE TABLE test(chartyp CHAR(10), varchartyp VARCHAR2(10));
INSERT INTO test VALUES('ab', 'cd');
SELECT chartyp, dump(chartyp), varchartyp, dump(varchartyp) ) FROM test;
```

CHARTYP	DUMP(CHARTYP)	VARCHARTYP	DUMP(VARCHARTYP)
1 ab	Typ=96 Len=10: 97,98,32,32,32,32,32,32,32,32 cd		Typ=1 Len=2: 99,100

**Warning:** Vorsicht Falle! Bis zur Version Oracle 11g lieferte die folgende Abfrage keine Tupel zurück.

```
SELECT * FROM test WHERE chartest='ab'
```

Frage:

- Weshalb wird bei Nutzung von Oracle 11g kein Tupel gefunden?
- Wie kann die SQL-Anfrage korrigiert werden, so dass das Tupel in Oracle 11g auch gefunden wird?
- Was liefert die aktuelle Oracle-Version für eine Ergebnismenge?

## Funktionsbeispiele

Funktion	Beispiel
UPPER(ausdruck)	UPPER('Anatomie-Atlas')= 'ANATOMIE-ATLAS'
LOWER(ausdruck)	LOWER('Anatomie-Atlas')= 'anatomie-atlas'
CONCAT (ausdruck1, ausdruck2) ausdruck1    ausdruck2	CONCAT(' 1', ' €')= ' 1'    ' €' = ' 1 €'
LENGTH(ausdruck)	LENGTH('Anatomie-Atlas')= 14
SUBSTR (ausdruck, von, laenge)	SUBSTR ('Anatomie-Atlas',10,5)= 'Atlas'
TRANSLATE (ausdruck, alt, neu)	TRANSLATE('Anatomie-Atlas','Atlas','Karte')= 'Anatomie-Karte'
TRIM(ausdruck)	TRIM(' Hallo ')= 'Hallo' pre- und post-Leerzeichen entfernt

### Beispiel

Gebe den Preis der Artikel zusammen mit der Währung in Euro aus.

### Mustervergleiche

- Zeichenkettenvergleich mit **LIKE**
  - % : kein oder beliebig viele Zeichen
  - \_ : für genau ein Zeichen
- Beispiele
  - Welche Namen beginnen mit ,Mei‘ ?

<b>SELECT</b>	Kundennummer, Nachname
<b>FROM</b>	Kunde
<b>WHERE</b>	Nachname <b>LIKE</b> 'Mei%'

- Mayer? Maier? Meier? Meyer? ...?

<b>SELECT</b>	Kundennummer, Nachname
<b>FROM</b>	Kunde
<b>WHERE</b>	UPPER(Nachname) <b>LIKE</b> 'M_ _ER%'

*Konvertierung in Großbuchstaben*

*Nachname CHAR(30)*

**Attention:** LIKE oder = verwenden?

- Für einen **Mustervergleich** ist der **LIKE**-Operator zu verwenden, hierbei wird ein Teil einer Zeichenkette ('Meier') mit einem *Muster* ('Mei%') verglichen.
- Sollen **alle** Zeichen übereinstimmen, dann ist der **Gleichheitsoperator** (=) zu verwenden.

## Reguläre Ausdrücke

Ein regulärer Ausdruck ist eine Zeichenfolge, mit der eine Reihe von Zeichenfolgen mithilfe von Syntaxregeln beschrieben werden.

- Der Beginn des regulären Ausdrucks wird durch das Zeichen ^ gekennzeichnet.
- Der reguläre Ausdruck wird mit dem Zeichen \$ abgeschlossen.

## Übung

Finde alle Kunden, deren Nachname entweder Maier, Meier, Mayer oder Meyer lautet.

### Spaghetti-SQL

```
SELECT Kundennummer, Nachname
FROM Kunde
WHERE TRIM(Nachname) LIKE 'Maier'
OR   TRIM(Nachname) LIKE 'Meier'
OR   TRIM(Nachname) LIKE 'Mayer'
OR   TRIM(Nachname) LIKE 'Meyer'
```

## Funktionen für Zeit- und Datumswerte

Beschreibung	Oracle	Beispiel
Datumsformat NLS_DATE_FORMAT	TT.MM.YYYY hh:mm:ss (Installationsabhängig!)	03.08.1997 02:52:17
Aktuelles Datum	SYSDATE, CURRENT_DATE CURRENT_TIMESTAMP	27.10.2023 27.10.2023 9:55:01,23 Berlin
Zeitzone	SESSIONTIMEZONE	EUROPE/Berlin
Extrahieren von Datumskomponenten	DATE; YEAR, MONTH, HOUR, TIMEZONE_ABBR,...	EXTRACT(YEAR FROM <i>GebDat</i> )
Datumsfunktionen (Auswahl)	ADD_MONTH MONTH_BETWEEN DATEDIFF ...	

### Beispiel

Welches Datum ist in 14 Tagen?

Beispiele äquivalenter Formulierungen in der WHERE-Bedingung:

Datum **>=** To\_Date('01-01-1875') **AND**  
Datum **<=** To\_Date('31-12-1879')

Datum <b>BETWEEN</b>	<small>(inkl. Intervallendpunkte)</small>
<u>To_Date('01-01-1875')</u> <b>AND</b> <u>To_Date('31-12-1879')</u>	

**EXTRACT (YEAR FROM** Datum) **BETWEEN** 1875 AND 1879

**EXTRACT (YEAR FROM** Datum)  
**IN** (1875, 1876, 1877, 1878, 1879)

*Datumsfunktion*

**Hint:** Änderung der Datums- und Zeitformate

SELECT * FROM NLS_SESSION_Parameters;	
Abfrageergebnis	Abfragen
Alle Zeilen abgerufen: 17 in 0,006 Sekunden	
PARAMETER	VALUE
1 NLS_LANGUAGE	GERMAN
2 NLS_TERRITORY	GERMANY
3 NLS_CURRENCY	€
4 NLS_ISO_CURRENCY	GERMANY
5 NLS_NUMERIC_CHARACTERS	,.
6 NLS_CALENDAR	GREGORIAN
7 NLS_DATE_FORMAT	DD.MM.RR
8 NLS_DATE_LANGUAGE	GERMAN
9 NLS_SORT	GERMAN
10 NLS_TIME_FORMAT	HH24:MI:SSXFF
11 NLS_TIMESTAMP_FORMAT	DD.MM.RR HH24:MI:SSXFF
12 NLS_TIME_TZ_FORMAT	HH24:MI:SSXFF TZR
13 NLS_TIMESTAMP_TZ_FORMAT	DD.MM.RR HH24:MI:SSXFF TZR
14 NLS_DUAL_CURRENCY	€
15 NLS_COMP	BINARY
16 NLS_LENGTH_SEMANTICS	BYTE
17 NLS_NCHAR_CONV_EXCP	FALSE

ALTER SESSION SET Ändern  
NLS\_DATE\_FORMAT = 'DD.MM.YYYY';

ALTER SESSION SET  
NLS\_DATE\_FORMAT = 'HH24:MI:SS';

ALTER SESSION SET  
NLS\_TIMESTAMP\_FORMAT = 'DD.MM.YYYY ...'

## Konvertierung von Datentypen

Oracle stellt verschiedene Konvertierungsfunktionen zwischen den Datentypen bereit.

Konvertierung	Beispiel
Zwischen Oracle-Standardtypen	CAST ('03.08.1997' as date) CAST(1110 as CHAR(5))
DATE → CHAR	TO_CHAR(SYSDATE, 'MM') = '11'
CHAR → DATE	TO_DATE('03.08.1997')

- 1) 

```
SELECT '*' || CAST(111 as CHAR(10)) || '*' FROM dual;
```

```
/*||CAST(111ASCHAR(10))||*/
*111 *
```
- 2) 

```
SELECT 'Zahlbar bis ' ||
       TO_CHAR (SYSDATE+14, 'DD.MM.YY') Zahlungfrist
  from dual;
```
- 3) 

```
SELECT TO_CHAR('0110'+1) FROM DUAL;
```

```
TO_CHAR('0110'+1)
111
```

Eine Konvertierung zwischen Datentypen ist mit der CAST-Funktion entsprechend der Verträglichkeitstabelle möglich.

### Beispiel

In die Tabelle Bestellung (Bestellnummer, Kundennummer, Bestellzeitpunkt) soll eine Bestellung mit der Bestellnummer 4711 und der Kundennummer 2310 für den 2.11.2023, um 18.12 Uhr, eingefügt werden. Das Attribut Bestellzeitpunkt ist als Timestamp definiert. Wie lautet das zugehörige Statement zum Einfügen der Bestellung?

---

### Selbsttest

Die folgenden Fragen sollten Sie jetzt beantworten können:

- Wie wird eine Anfrage aus der relationalen Algebra in SQL umgeschrieben?
- Wie ist eine SQL-Abfrage aufgebaut?
- Welche Funktionen können in Oracle verwendet werden?

### Aufgabe 1

Erläutern Sie, was die folgende Anfrage macht.

Schreiben Sie die Anfrage in SQL um.

```
ρ ID←Kundennummer, Wohnort←Ort  π Kundennummer, Nachname, Ort  
σ Anrede = 'Herr' Kunde
```

### Aufgabe 2

Es sollen alle Geschäftskunden aus Dortmund in der Form Kundennummer und Name gelistet werden. Formulieren Sie die dafür notwendigen Befehle in der relationalen Algebra **und** in SQL.

### Aufgabe 3

Es sollen alle Artikel mit Artikelnummer und Artikelname gelistet werden, denen ein Lagerplatz zugeordnet ist. Formulieren Sie die dafür notwendigen Befehle in der relationalen Algebra **und** in SQL.