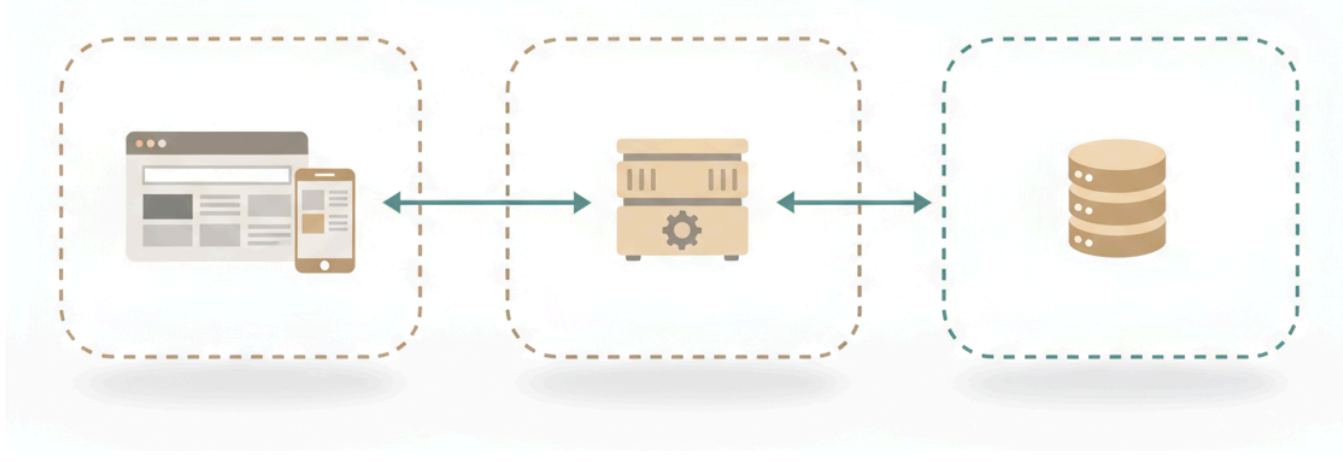
Datengewinnung und -analyse

2. Datenbanken und SQL

Markus Schanta

Architektur einer Webapplikation



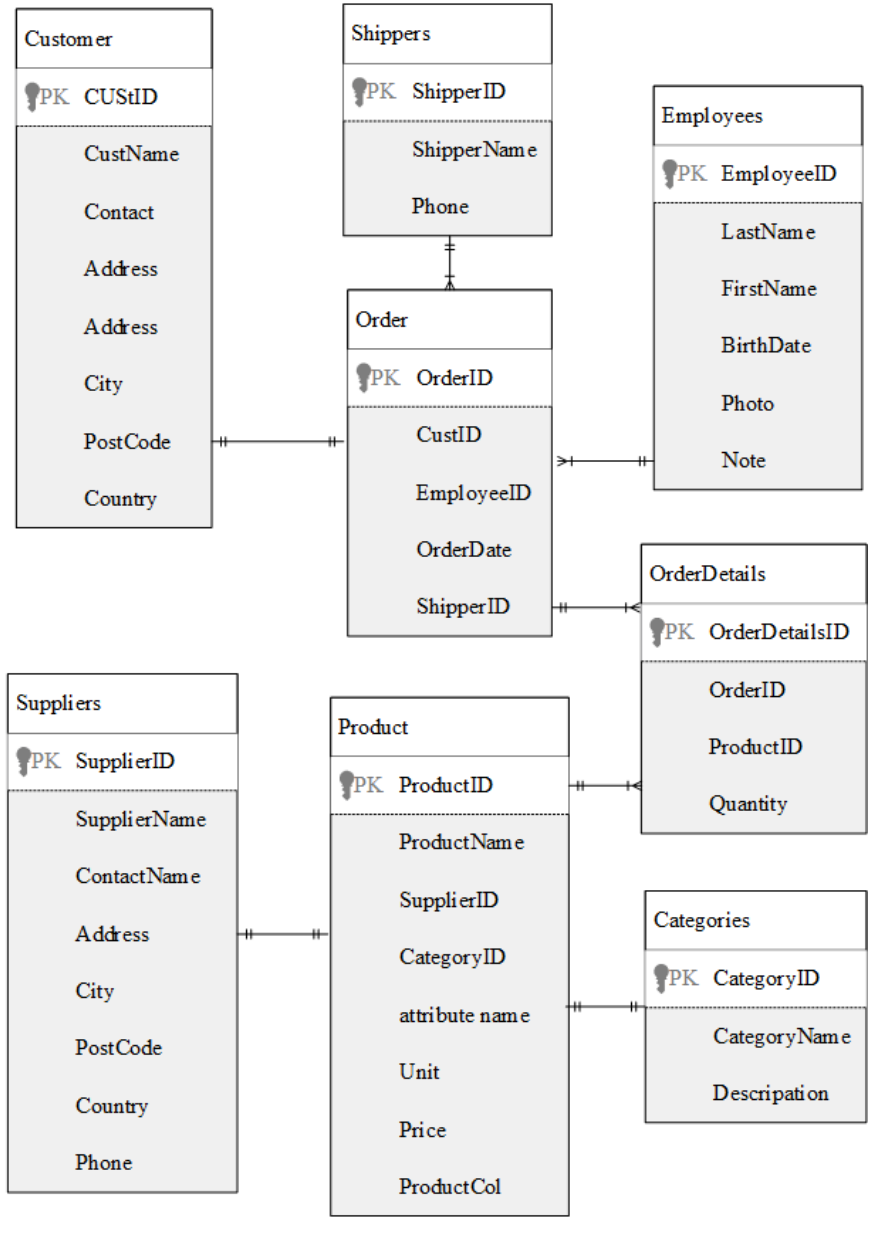
Komponenten einer Datenbank

Aufbau einer Tabelle

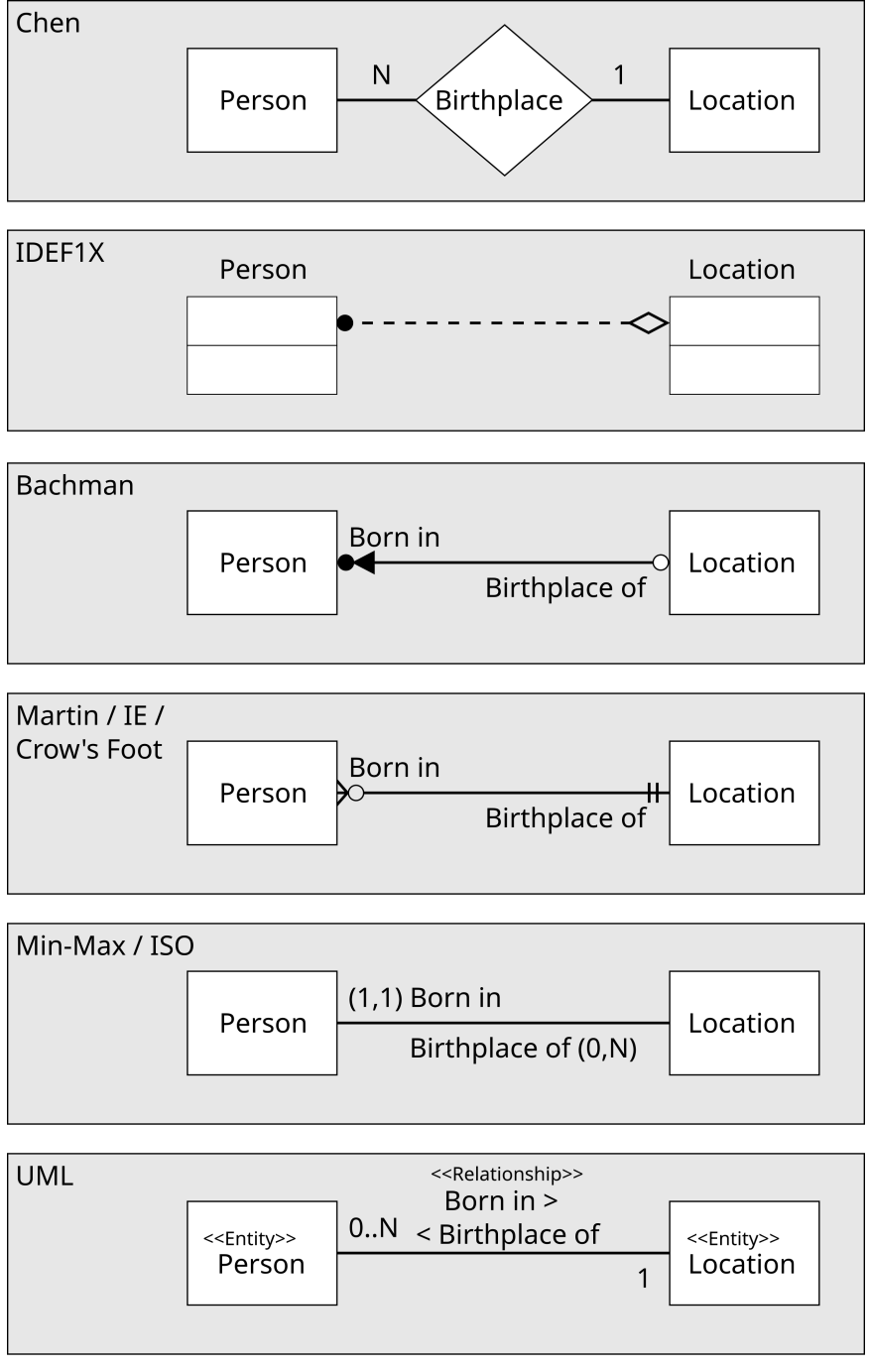
Number of Records: 77

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

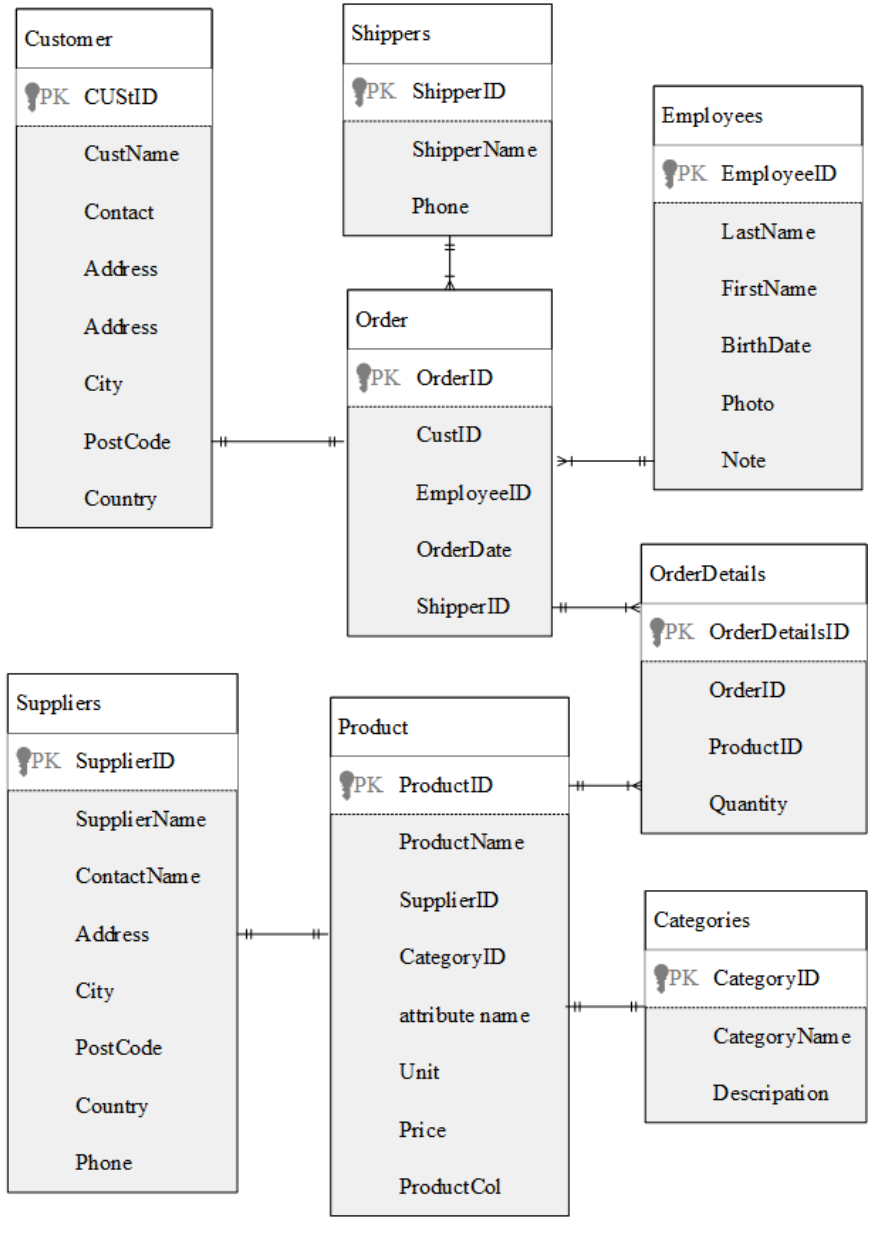
Entity-Relationship Diagramm



ER Notationen


Chen
Person
N
Birthplace
1
Location
IDEF1X
Person
Location
Bachman
Person
Born in
Birthplace of
Location
Martin / IE /
Crow's Foot
Person
Born in
Birthplace of
Location
Min-Max / ISO
Person
(1,1) Born in
Birthplace of (0,N)
Location
UML
<<Relationship>>
Born in >
< Birthplace of
<<Entity>>
Person
0..N
1
<<Entity>>
Location

Primary Key / Foreign Key



Der Begriff "Datenbank"

bezeichnet

Der Begriff "Datenbank"

bezeichnet

1. eine strukturierte Sammlung von Daten (= Schema + Tabelleninhalte) und

Der Begriff "Datenbank"

bezeichnet

1. eine strukturierte Sammlung von Daten (= Schema + Tabelleninhalte) und
2. Mechanismen und Protokolle zur Verwaltung dieser Daten.

SQL

SQL

- **SQL** ist ein Akronym für **S**tructured **Q**uery **L**anguage.

SQL

- **SQL** ist ein Akronym für **S**tructured **Q**uery **L**anguage.
- Erstmals 1974 veröffentlicht, seit den 1990ern als Standard etabliert.

SQL

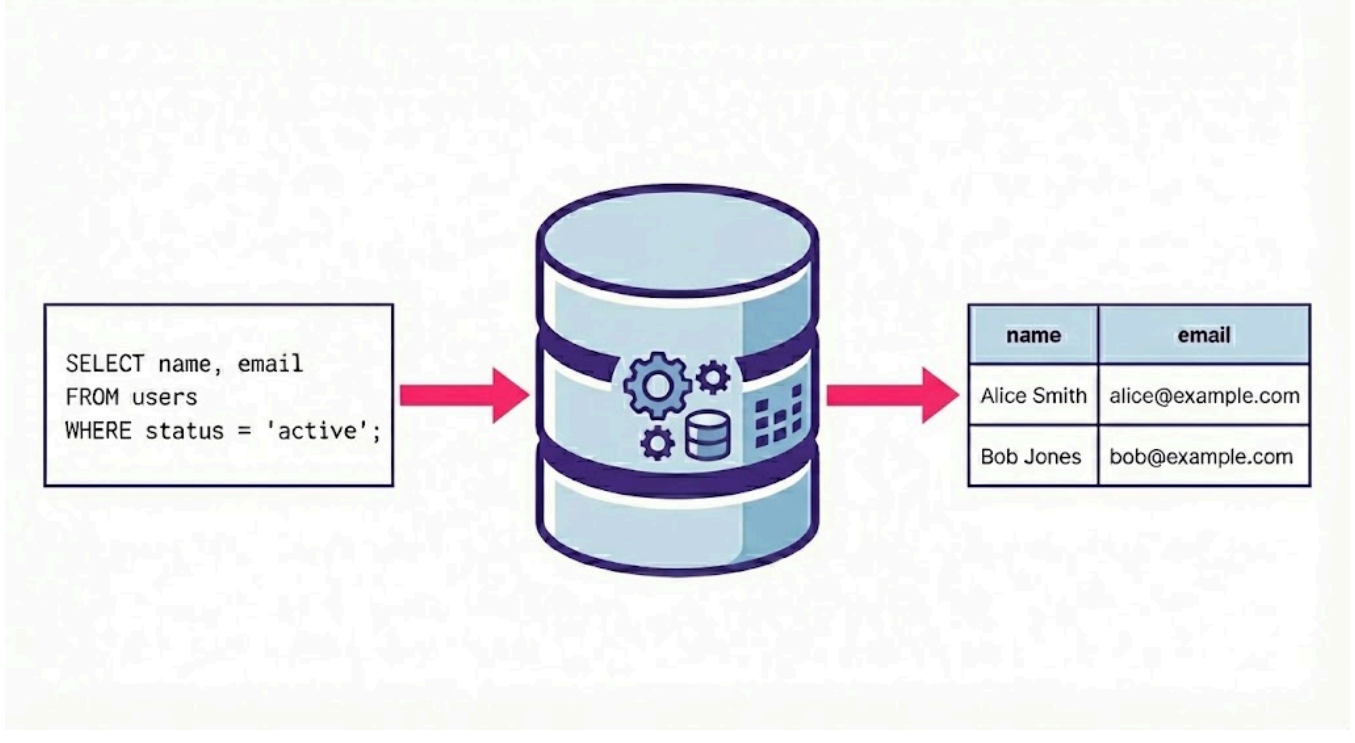
- **SQL** ist ein Akronym für **S**tructured **Q**uery **L**anguage.
- Erstmals 1974 veröffentlicht, seit den 1990ern als Standard etabliert.
- Vier Hauptfunktionen:
    1. Create (Schlüsselwort `INSERT`)
    2. **Retreive (Schlüsselwort `SELECT`)**
    3. Update (Schlüsselwort `UPDATE`)
    4. Delete (Schlüsselwort `DELETE`)

SQL: Ausgangspunkt ist eine Tabelle

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Request → Response



SELECT-Abfragen

SELECT-Abfragen

- Grundstruktur einer SQL-Abfrage:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
```

SELECT-Abfragen

- Grundstruktur einer SQL-Abfrage:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
```

- **SELECT** gibt an, welche Spalten ausgewählt werden

SELECT-Abfragen

- Grundstruktur einer SQL-Abfrage:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
```

- **SELECT** gibt an, welche Spalten ausgewählt werden
- **FROM** gibt an, aus welcher Tabelle die Daten stammen

SELECT-Abfragen

- Grundstruktur einer SQL-Abfrage:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
```

- **SELECT** gibt an, welche Spalten ausgewählt werden
- **FROM** gibt an, aus welcher Tabelle die Daten stammen
- Spalte `CustomerName` aus der Tabelle `Customers` auswählen:

```
SELECT CustomerName
FROM Customers
```

SELECT-Abfragen

- Grundstruktur einer SQL-Abfrage:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
```

- **SELECT** gibt an, welche Spalten ausgewählt werden
- **FROM** gibt an, aus welcher Tabelle die Daten stammen
- Spalte `CustomerName` aus der Tabelle `Customers` auswählen:

```
SELECT CustomerName
FROM Customers
```

- Alle Spalten aus der Tabelle `Customers` auswählen:

```
SELECT *
FROM Customers
```

W3Schools SQL Editor

SQL Statement:

Get your own SQL server

```
SELECT CategoryID
FROM Products
GROUP BY CategoryID;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 8

CategoryID
1
2
3
4

Your Database:

Tablenames	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

SELECT Beispiel 1

- Die Spalte `ContactName` aus der Tabelle `Customers` auswählen:

SELECT Beispiel 1

- Die Spalte `ContactName` aus der Tabelle `Customers` auswählen:

```
SELECT ContactName
FROM Customers
```

SELECT Beispiel 2

- Die Spalten `CustomerName` und `ContactName` aus der Tabelle `Customers` auswählen:

SELECT Beispiel 2

- Die Spalten `CustomerName` und `ContactName` aus der Tabelle `Customers` auswählen:

```
SELECT CustomerName, ContactName
FROM Customers
```

SELECT Beispiel 3

- Die Postleitzahl und Stadt aller Kunden ausgeben:

SELECT Beispiel 3

- Die Postleitzahl und Stadt aller Kunden ausgeben:

```
SELECT PostalCode, City
FROM Customers
```

SELECT Beispiel 4

- Alle Informationen zu allen Kunden ausgeben:

SELECT Beispiel 4

- Alle Informationen zu allen Kunden ausgeben:

```
SELECT *
FROM Customers
```

SELECT Zusammenfassung

- Die Spalte `ContactName` auswählen:

```
SELECT ContactName
FROM Customers
```

- Die Spalten `CustomerName` und `ContactName` ausgeben:

```
SELECT CustomerName, ContactName
FROM Customers
```

- Alle Spalten ausgeben:

```
SELECT *
FROM Customers
```

Abfragen mit WHERE-Bedingungen

- Grundlegende Struktur einer **SELECT**-Abfrage mit **WHERE**-Filter:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
WHERE bedingung1
    AND/OR bedingung2
    AND/OR ...;
```

Operatoren für `WHERE`-Bedingungen

Operator	Bedingung	Beispiel
`=, !=, <, <=, >, >=.`	Numerische Operatoren	`spalte > 4`
`BETWEEN ... AND ...`	Zahl zwischen zwei Werten	`spalte BETWEEN 1.5 AND 10.5`
`NOT BETWEEN ... AND ...`	Zahl nicht zwischen Werten	`spalte NOT BETWEEN 1 AND 10`
`IN (...)`	Zahl ist in einer Liste	`spalte IN (2, 4, 6)`
`NOT IN (...)`	Zahl ist nicht in einer Liste	`spalte NOT IN (1, 3, 5)`

`WHERE`-Bedingungen Beispiel 1

- Gib die `Namen` und `Preise` aller `Produkte` mit einem `Preis` von mehr als `50` aus:

`WHERE`-Bedingungen Beispiel 1

- Gib die `Namen` und `Preise` aller `Produkte` mit einem `Preis` von mehr als `50` aus:

```
SELECT ProductName, Price
FROM Products
WHERE Price > 50
```

`WHERE`-Bedingungen Beispiel 2

- Gib die `Namen` aller `Produkte` mit einem `Preis` zwischen `10` und `13` aus:

`WHERE`-Bedingungen Beispiel 2

- Gib die `Namen` aller `Produkte` mit einem `Preis` zwischen `10` und `13` aus:

```
SELECT ProductName
FROM Products
WHERE Price BETWEEN 10 AND 13
```

`WHERE`-Bedingungen Beispiel 2

- Gib die `Namen` aller `Produkte` mit einem `Preis` zwischen `10` und `13` aus:

```
SELECT ProductName
FROM Products
WHERE Price BETWEEN 10 AND 13
```

- *alternativ*

WHERE-Bedingungen Beispiel 2

- Gib die `Namen` aller `Produkte` mit einem `Preis` zwischen `10` und `13` aus:

```
SELECT ProductName
FROM Products
WHERE Price BETWEEN 10 AND 13
```

- *alternativ*

```
SELECT ProductName
FROM Products
WHERE Price >= 10 AND Price <= 13
```

Spalten / Zeilen

- **SELECT** bestimmt die **Spalten**, **WHERE** die :

```
SELECT spalte1, spalte2, ...
FROM tabellenname
WHERE bedingung
```

Result:

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany

Spalten / Zeilen

- **SELECT** bestimmt die **Spalten**, **WHERE** die **Zeilen**:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
WHERE bedingung
```

Result:

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany

WHERE: Operatoren für Strings

Operator	Bedingung	Beispiel
`=`	Exakte Übereinstimmung (case sensitive)	`col_name = "abc"`
`!=` oder `<>`	Ungleichheit	`col_name != "abcd"`
`LIKE`	Übereinstimmung (case insensitive)	`col_name LIKE "ABC"`
`NOT LIKE`	Nicht-Übereinstimmung	`col_name NOT LIKE "ABCD"`
`%`	Platzhalter für mehrere (oder keine) Zeichen	`col_name LIKE "%AT%"` (findet "AT", "ATTIC", "CAT" oder "BATS")
`_`	Platzhalter für genau ein Zeichen	`col_name LIKE "AN_"` (findet “AND“ aber nicht “AN")

`WHERE`-Bedingungen Beispiel 3

- Finde alle `Kunden` in `Deutschland`:

WHERE-Bedingungen Beispiel 3

- Finde alle Kunden in Deutschland:

```
SELECT *
FROM Customers
WHERE Country = 'Germany'
```

`WHERE`-Bedingungen Beispiel 4

- Finde alle Kunden in Ländern, die mit „land“ enden:

WHERE-Bedingungen Beispiel 4

- Finde alle Kunden in Ländern, die mit „land“ enden:

```
SELECT *
FROM Customers
WHERE Country LIKE '%land'
```

`WHERE`-Bedingungen Beispiel 5

- Finde alle `Kunden` in `Ländern` mit einer vierstelligen `Postleitzahl`:

WHERE-Bedingungen Beispiel 5

- Finde alle Kunden in Ländern mit einer vierstelligen `Postleitzahl`:

```
SELECT *
FROM Customers
WHERE PostalCode LIKE '____'
```

SQL-Abfragen in Google Colab

Sortierung mit `ORDER BY`

Sortierung mit `ORDER BY`

- Sortiert das Ergebnis nach der angegebenen Spalte:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
WHERE bedingung
ORDER BY spalte1 [ASC|DESC]
```

Sortierung mit `ORDER BY`

- Sortiert das Ergebnis nach der angegebenen Spalte:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
WHERE bedingung
ORDER BY spalte1 [ASC|DESC]
```

- Standardmäßig aufsteigend (**`ASC`**)

Sortierung mit `ORDER BY`

- Sortiert das Ergebnis nach der angegebenen Spalte:

```
SELECT spalte1, spalte2, ...
FROM tabellenname
WHERE bedingung
ORDER BY spalte1 [ASC|DESC]
```

- Standardmäßig aufsteigend (**`ASC`**)
- Absteigend mit **`DESC`**

`ORDER BY` Beispiel 1

- Gib alle `Produkte` sortiert nach `Preis` aus:

`ORDER BY` Beispiel 1

- Gib alle `Produkte` sortiert nach `Preis` aus:

```
SELECT *
FROM Products
ORDER BY Price ASC
```

De-Duplizierung mit `DISTINCT`

De-Duplizierung mit `DISTINCT`

- Entfernt doppelte Einträge aus dem Ergebnis:

```
SELECT DISTINCT spalte1
FROM tabellenname
WHERE bedingung
```

Gruppierung & Aggregatfunktionen

Gruppierung & Aggregatfunktionen

- Gruppierung mit **`GROUP BY`**:

Gruppierung & Aggregatfunktionen

- Gruppierung mit **GROUP BY**:

```
SELECT CategoryID
FROM Products
GROUP BY CategoryID
```

Gruppierung & Aggregatfunktionen

Gruppierung & Aggregatfunktionen

- Gruppierung mit `GROUP BY` und Aggregatfunktion `COUNT()`:

Gruppierung & Aggregatfunktionen

- Gruppierung mit **`GROUP BY`** und Aggregatfunktion **`COUNT()`**:

```sql
SELECT CategoryID, COUNT(ProductID)
FROM Products
GROUP BY CategoryID
```

Aggregatfunktionen in SQL

Funktion	Bedeutung
`COUNT(*)`	Anzahl aller Zeilen
`COUNT(spalte)`	Anzahl nicht-null Werte in `spalte`
`MIN(spalte)`	Kleinster Wert in `spalte`. `spalte` muss numerisch sein.
`MAX(spalte)`	Größter Wert in `spalte`. `spalte` muss numerisch sein.
`AVG(spalte)`	Durchschnittlicher Wert in `spalte`. `spalte` muss numerisch sein.
`SUM(spalte)`	Summe aller Werte in `spalte`. `spalte` muss numerisch sein.

`GROUP BY` Beispiel 1

- Gib den durchschnittlichen `Preis` pro `Produktkategorie` aus:

GROUP BY Beispiel 1

- Gib den durchschnittlichen `Preis` pro `Produktkategorie` aus:

```
SELECT CategoryID, AVG(Price)
FROM Products
GROUP BY CategoryID
```

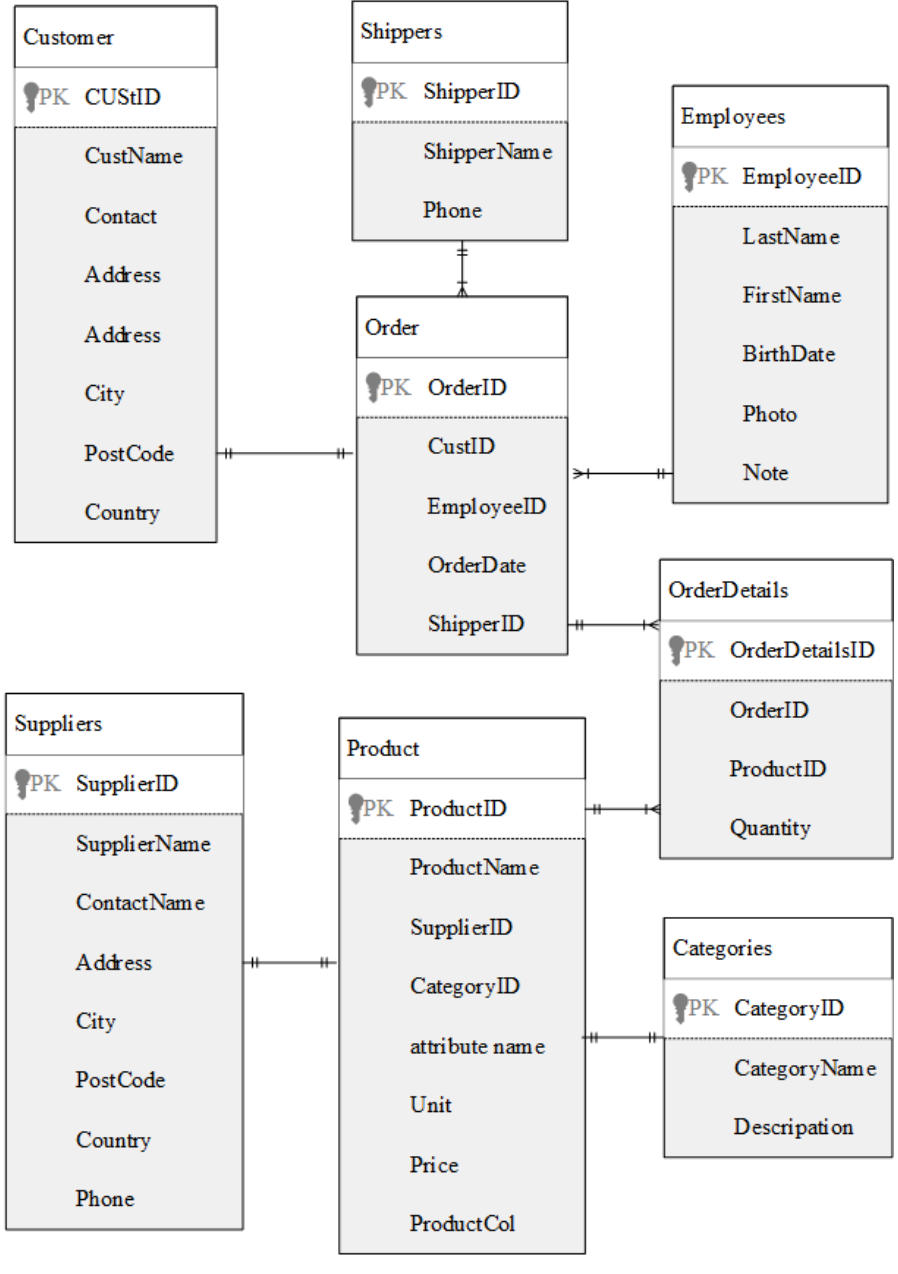
Normalisierung

Number of Records: 77

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

Number of Records: 8

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish


Customer
PK CUStID
CustName
Contact
Address
Address
City
PostCode
Country
Shippers
PK ShipperID
ShipperName
Phone
Employees
PK EmployeeID
LastName
FirstName
BirthDate
Photo
Note
Order
PK OrderID
CustID
EmployeeID
OrderDate
ShipperID
OrderDetails
PK OrderDetailsID
OrderID
ProductID
Quantity
Suppliers
PK SupplierID
SupplierName
ContactName
Address
City
PostCode
Country
Phone
Product
PK ProductID
ProductName
SupplierID
CategoryID
attribute name
Unit
Price
ProductCol
Categories
PK CategoryID
CategoryName
Descripation

Joins von Tabellen

Joins von Tabellen

- **`LEFT JOIN`** der Tabellen `Products` und `Categories`:

Joins von Tabellen

- **`LEFT JOIN`** der Tabellen `Products` und `Categories`:

```
SELECT *
FROM Products
LEFT JOIN Categories
  ON Products.CategoryID = Categories.CategoryID
```

Joins von Tabellen

- **LEFT JOIN** der Tabellen `Products` und `Categories`:

```
SELECT *
FROM Products
LEFT JOIN Categories
  ON Products.CategoryID = Categories.CategoryID
```

Number of Records: 77

ProductID	ProductName	SupplierID	Products.CategoryID	Unit	Price	Categories.CategoryID	CategoryName	Description
1	Chais	1	1	10 boxes x 20 bags	18	1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Chang	1	1	24 - 12 oz bottles	19	1	Beverages	Soft drinks, coffees, teas, beers, and ales
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22	2	Condiments	Sweet and savory sauces, relishes, spreads, and

JOIN Beispiel 1

`JOIN` Beispiel 1

- Gib alle `Produkte` aus der `Produktkategorie Produce` aus:

JOIN Beispiel 1

- Gib alle `Produkte` aus der `Produktkategorie Produce` aus:

```
SELECT *
FROM Products
LEFT JOIN Categories
  ON Products.CategoryID = Categories.CategoryID
WHERE CategoryName='Produce'
```

JOIN Beispiel 1

- Gib alle Produkte aus der Produktkategorie Produce aus:

```
SELECT *
FROM Products
LEFT JOIN Categories
  ON Products.CategoryID = Categories.CategoryID
WHERE CategoryName='Produce'
```

Number of Records: 5

ProductID	ProductName	SupplierID	Products.CategoryID	Unit	Price	Categories.CategoryID	CategoryName	Description
14	Tofu	6	7	40 - 100 g pkgs.	23.25	7	Produce	Dried fruit and bean curd
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30	7	Produce	Dried fruit and bean curd
28	Rössle Sauerkraut	12	7	25 - 825 g cans	45.6	7	Produce	Dried fruit and bean curd
74	Longlife Tofu	4	7	5 kg pkg.	10	7	Produce	Dried fruit and bean curd
51	Manjimup Dried Apples	24	7	50 - 300 g pkgs.	53	7	Produce	Dried fruit and bean curd

JOIN Beispiel 2

`JOIN` Beispiel 2

- Gib die Anzahl der `Produkte` pro `Produktkategorie` aus:

JOIN Beispiel 2

- Gib die Anzahl der `Produkte` pro `Produktkategorie` aus:

```
SELECT CategoryName, COUNT(ProductID) as ProductIDCount
FROM Products
LEFT JOIN Categories
  ON Products.CategoryID = Categories.CategoryID
GROUP BY CategoryName
```

JOIN Beispiel 2

- Gib die Anzahl der Produkte pro Produktkategorie aus:

```sql
SELECT CategoryName, COUNT(ProductID) as ProductIDCount
FROM Products
LEFT JOIN Categories
  ON Products.CategoryID = Categories.CategoryID
GROUP BY CategoryName
```

Number of Records: 8

CategoryName	ProductIDCount
Beverages	12
Condiments	12
Confections	13
Dairy Products	10
Grains/Cereals	7
Meat/Poultry	6
Produce	5
Seafood	12

JOIN Beispiel 3

`JOIN` Beispiel 3

- `JOIN` über mehrere Tabellen: Gib alle `Produkte` mit `Lieferantenname` und `Produktkategorie` aus:

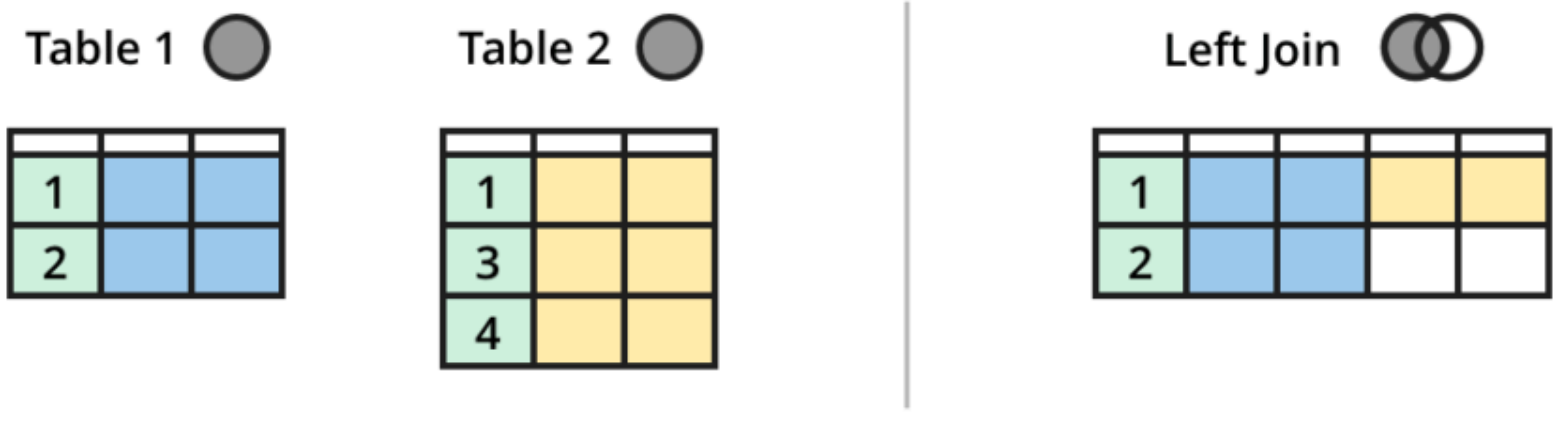
JOIN Beispiel 3

- JOIN über mehrere Tabellen: Gib alle Produkte mit Lieferantenname und Produktkategorie aus:

```
SELECT *
FROM Products p
LEFT JOIN Categories c
  ON p.CategoryID=c.CategoryID
LEFT JOIN Suppliers s
  ON p.SupplierID=s.SupplierID
```

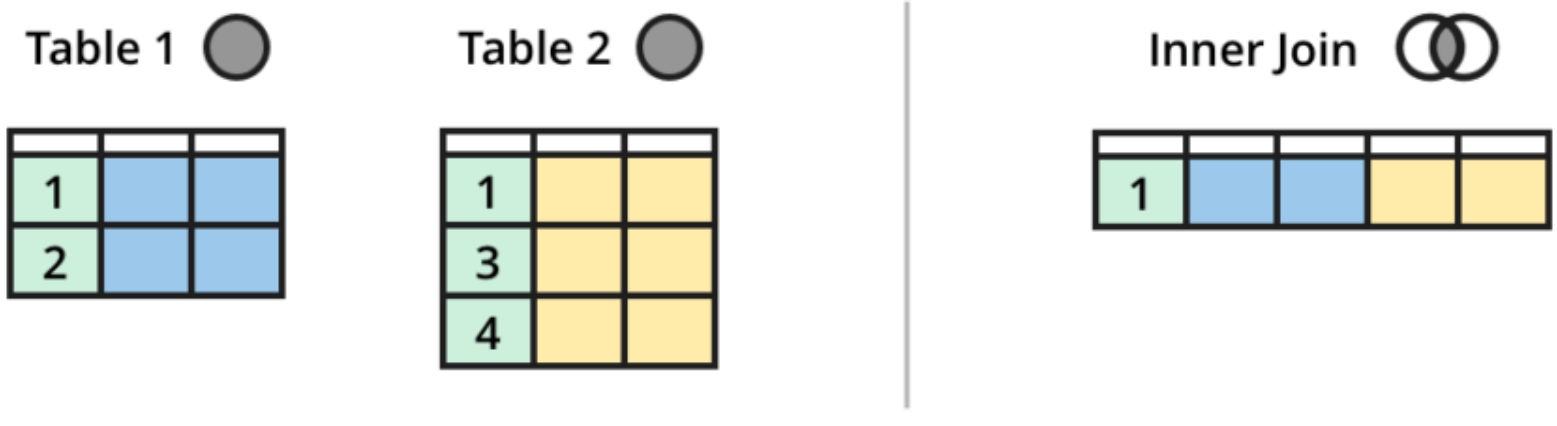
`LEFT JOIN` Bedeutung

`LEFT JOIN` bedeutet, dass alle Zeilen der linken Tabelle im Ergebnis enthalten sind, Zeilen der rechten Tabelle aber nur, wenn es einen passenden Eintrag gibt.



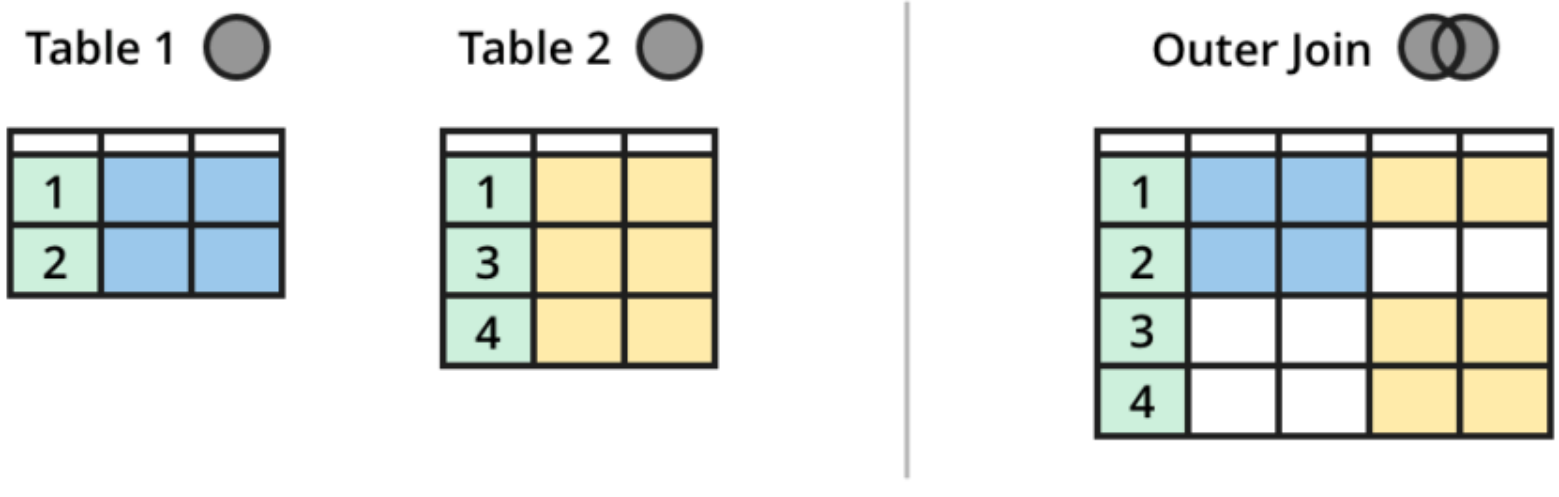
`INNER JOIN` Bedeutung

`INNER JOIN` bedeutet, dass ausschließlich, die Zeilen der linken und rechten Tabelle im Ergebnis enthalten sind, die übereinstimmende Werte in den verbundenen Spalten haben.



`OUTER JOIN` Bedeutung

`OUTER JOIN` bedeutet, dass alle Zeilen der linken und rechten Tabelle im Ergebnis enthalten sind und übereinstimmende Werte im Ergebnis in einer Zeile zusammengeführt werden.



Fragen?

Nächste Schritte

- Folien & Materialien der heutigen Einheit auf Moodle
- Nächste Einheit: Do. 4.12. 18:30

Danke und schönen Abend!