# Themen


| | Seite | |
|---|---|---|
| Sortieren und Filtern in Tabellen | 10 | 1 |
| Mit Formeln und Funktionen arbeiten | 13 | 2 |
| Daten graphisch darstellen | 16 | 3 |
| Sensoren und Aktoren im Smartphone | 21 | 4 |
| Pocket Code App | 24 | 5 |
| Sensoren und Aktoren ansteuern | 28 | 6 |
| Sensordaten auslesen | 33 | 7 |
| Sensordaten auswerten | 37 | 8 |

weitere Themen siehe Seite 7

# Themen


| | Seite | |
|---|---|---|
| Datenkompression und Datenreduktion | 39 | 9 |
| Lauflängencodierung | 41 | 10 |
| Huffman-Codierung | 45 | 11 |
| Auflösung und Farbtiefe | 50 | 12 |
| Samplingrate und Samplingtiefe | 53 | 13 |
| Grafikformate GIF, PNG und JPG | 56 | 14 |
| Audioformat MP3 | 63 | 15 |
| Datenbanksysteme | 65 | 16 |
| Datenmodellierung | 67 | 17 |
| Relationales Datenbankschema | 70 | 18 |
| Datenbanktabellen mit SQL | 74 | 19 |
| Daten abfragen mit SQL | 77 | 20 |


weitere Themen siehe Seite 9

# Themen


| | Seite | |
|---|---|---|
| One-Time-Pad-Verfahren (OTP) | 84 | 21 |
| Advanced Encryption Standard (AES) | 87 | 22 |
| Ende-zu-Ende-Verschlüsselung | 91 | 23 |
| RSA-Verschlüsselung | 93 | 24 |
| Nachrichten signieren | 96 | 25 |
| Kerckhoffs' Prinzip | 98 | 26 |
| Hypertext Transfer Protocol Secure (HTTPS) | 100 | 27 |

# Daten graphisch darstellen

Daten lassen sich in Tabellen kompakt und übersichtlich zusammenfassen. Doch umfangreiche Tabellen sind mühsam zu lesen und häufig ist es schwer, Zusammenhänge zwischen den Werten zu erkennen.

Diagramme helfen, die Tabellendaten anschaulich darzustellen, so dass Verläufe, Trends und Größenvergleiche leichter zu erfassen sind.

Tabellenkalkulations-Software unterstützt die Darstellung der Tabellendaten in Diagrammform durch vordefinierte Diagrammtypen.

Im Menü „Einfügen“ gelangt man – je nach Software – über eines dieser Symbole zu einem Menü, das bei der Auswahl eines geeigneten Diagrammtyps hilft.

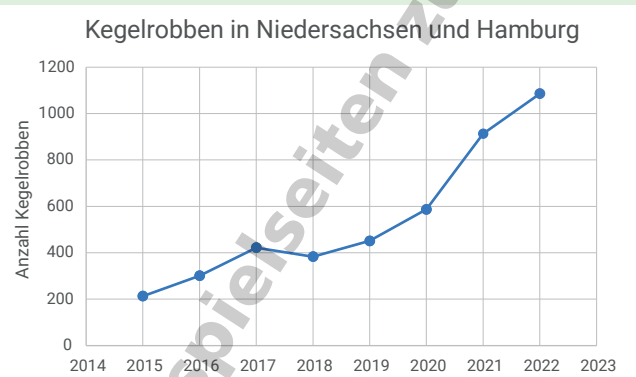

Empfohlene Diagramme


Diagramme

| Jahr | Kegelrobben [1] |
|---|---|
| **2015** | 213 |
| **2016** | 301 |
| **2017** | 422 |
| **2018** | 383 |
| **2019** | 451 |
| **2020** | 587 |
| **2021** | 913 |
| **2022** | 1086 |

Für die Darstellung der Entwicklung des Kegelrobben-Bestandes im Wattenmeer Niedersachsens und Hamburgs bietet sich beispielsweise ein XY-Diagramm an, das auch Streu- oder Punktdiagramm genannt wird.

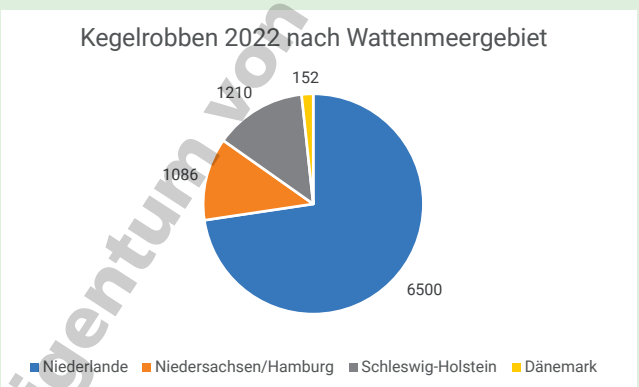

Im Diagramm-Assistenten oder über dieses Symbol lassen sich weitere Diagrammelemente wie eine Legende, Achsenbeschriftungen oder eine Überschrift zum Diagramm hinzufügen.




Das Vergleichen unterschiedlicher Mengen gelingt am anschaulichsten mit Kreisdiagrammen.

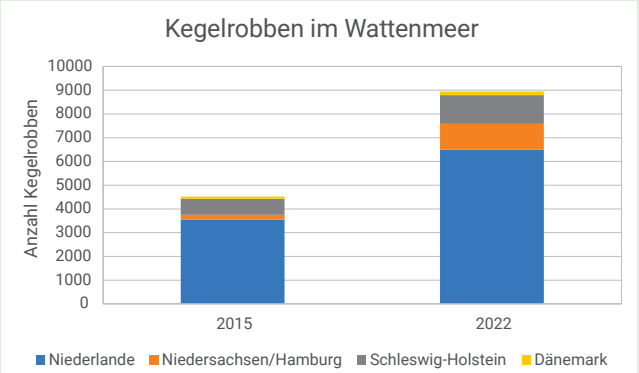
| Wattenmeergebiet | Kegelrobben [1] 2015 | 2022 |
|---|---|---|
| Niederlande | 3544 | 6500 |
| Niedersachsen/Hamburg | 213 | 1086 |
| Schleswig-Holstein | 676 | 1210 |
| Dänemark | 88 | 152 |

Dieses Kreisdiagramm zeigt beispielsweise auf Anhieb, dass fast drei Viertel der Kegelrobben im niederländischen Teil des Wattenmeeres leben.




In Kreisdiagrammen sind die Daten immer als Teile eines Ganzen, also in prozentualer Verteilung dargestellt. Es eignet sich daher nicht, um die in unterschiedlichen Jahren im Wattenmeer gezählten Kegelrobben miteinander zu vergleichen.

So genannte gestapelte Säulendiagramme zeigen sowohl die Mengenverteilung innerhalb jeder Säule als auch den Vergleich der Gesamtmengen anhand der Säulenhöhe. Sie bieten sich daher an, die 2015 und 2022 im Wattenmeer gezählten Kegelrobben miteinander zu vergleichen.






[1] Monitoring-Ergebnisse der Kegelrobbenzählungen im Wattenmeer in Niedersachsen und Hamburg https://www.waddensea-secretariat.org/de/seehunde (Stand April 2023)

# Daten graphisch darstellen

**Aufgabe 1**

Öffne die Datei Temperatur_Hameln_02-2023.xlsx [1].

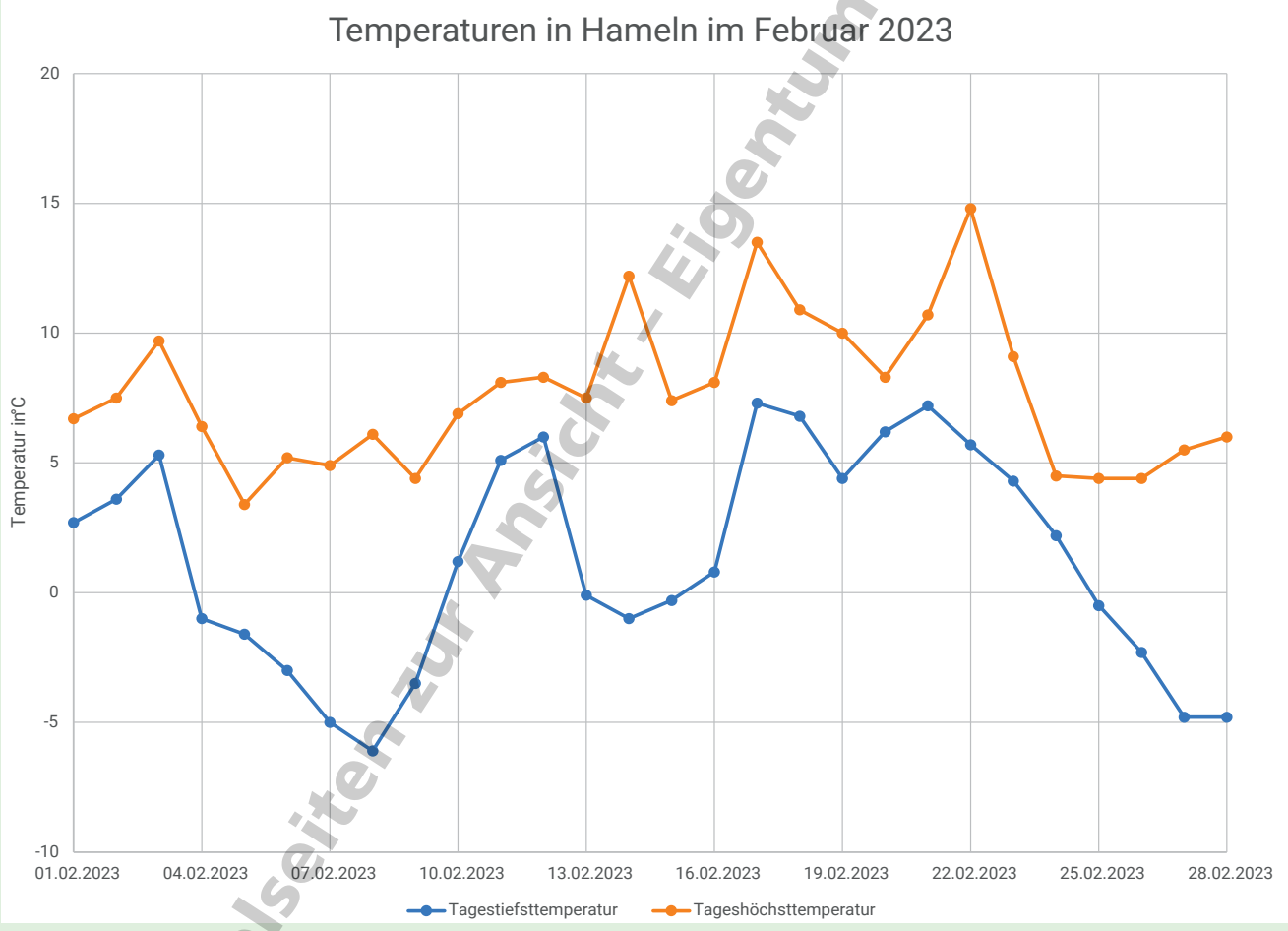
a) Erstelle ein XY-Diagramm, das die Tagestiefst- und die Tageshöchsttemperaturen für Hameln im Februar 2023 darstellt.

b) Füge eine Überschrift und einen Achsentitel an der Y-Achse ein.

Zusatzaufgaben

c) Formatiere die Y-Achse so, dass die horizontale Achse sie bei –10 °C schneidet.

d) Formatiere die X-Achse so, dass sie mit dem 1. Februar beginnt und mit dem 28. Februar endet.
Beachte dabei, dass Microsoft Excel anstelle des Datums den so genannten DATWERT anzeigt. Das ist eine Funktion, die vom 1. Januar 1900 (DATWERT = 1) ausgehend fortlaufend die Tage zählt. Der 1. Februar 2023 hat den DATWERT 44958 und der 28. Februar den DATWERT 44985.






Datenquellen:

[1] https://meteostat.net/de/place/de/hameln?s=D3675&t=2023-02-01/2023-02-28 (Stand April 2023)

[2] https://www1.nls.niedersachsen.de/statistik/html/default.asp; 12411 - Fortschreibung des Bevölkerungsstandes; Bevölkerung nach Altersgruppen (23) und Geschlecht (Gemeinde) (Stand Februar 2023)

[3] Flächenerhebung nach Art der tatsächlichen Nutzung, https://www.statistik.niedersachsen.de/flaechenerhebung/flachenerhebung-nach-art-der-tatsachlichen-nutzung-statistische-berichte-87671.html (Stand April 2023)

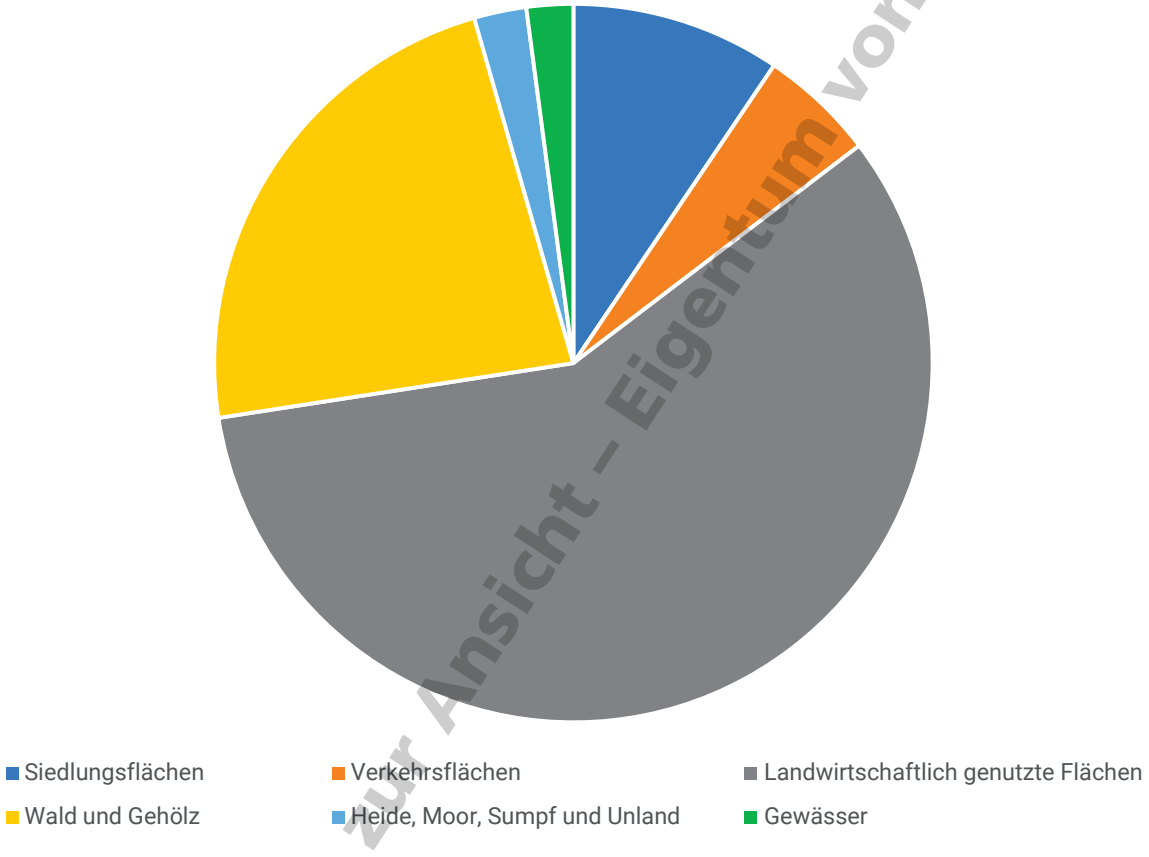
# Daten graphisch darstellen

**Aufgabe 2**

Öffne die Datei Flaechennutzung_NI.xlsx [3], die Daten zur Nutzung von Flächen in Niedersachsen enthält.

a) Erstelle ein Kreisdiagramm für die Flächennutzung in Niedersachsen.

b) Füge eine Überschrift und eine Legende hinzu.






Datenquellen:

[1] https://meteostat.net/de/place/de/hameln?s=D3675&t=2023-02-01/2023-02-28 (Stand April 2023)

[2] https://www1.nls.niedersachsen.de/statistik/html/default.asp; 12411 - Fortschreibung des Bevölkerungsstandes; Bevölkerung nach Altersgruppen (23) und Geschlecht (Gemeinde) (Stand Februar 2023)

[3] Flächenerhebung nach Art der tatsächlichen Nutzung, https://www.statistik.niedersachsen.de/flaechenerhebung/flachenerhebung-nach-art-der-tatsachlichen-nutzung-statistische-berichte-87671.html (Stand April 2023)

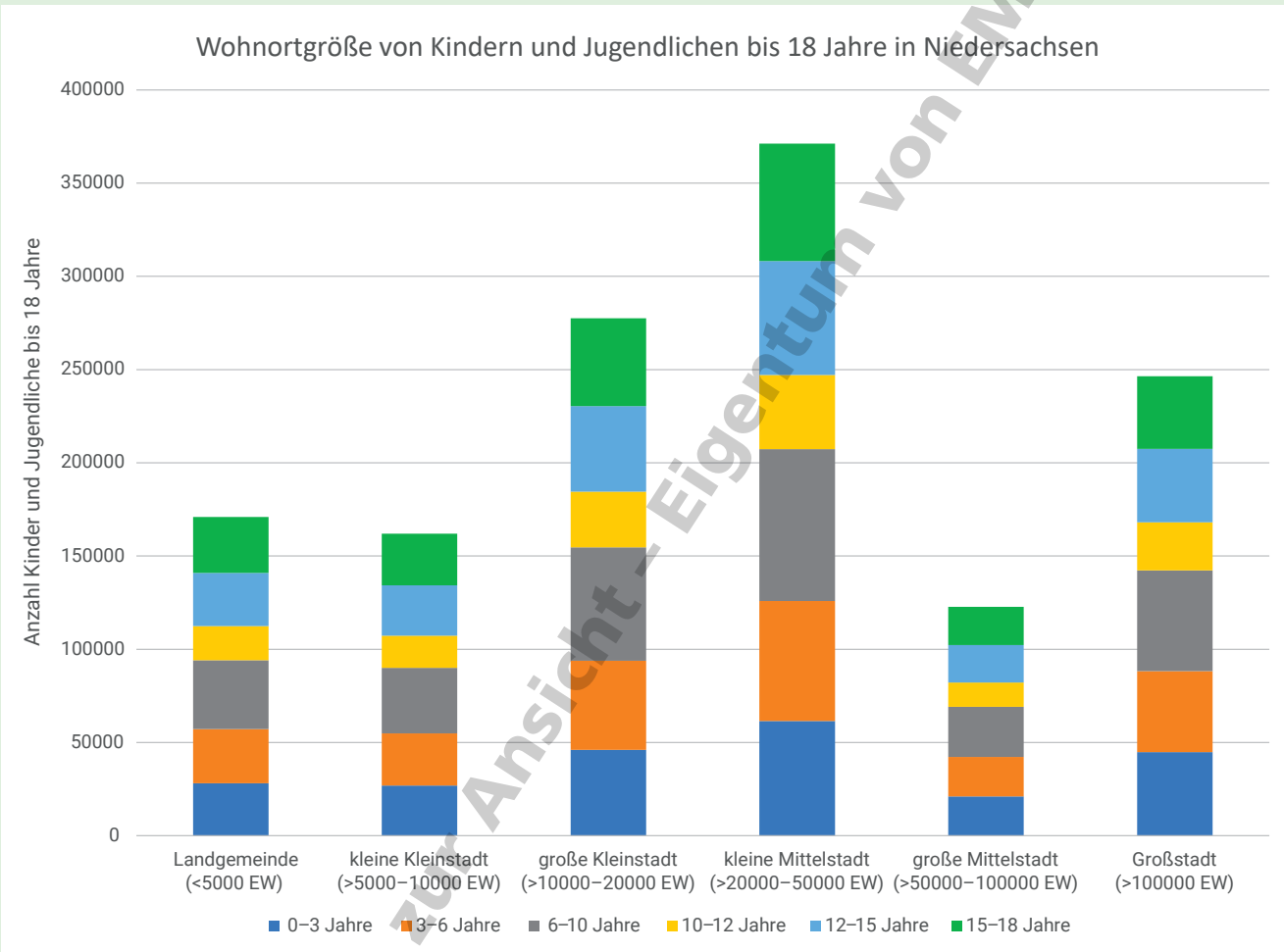
# Daten graphisch darstellen

**Aufgabe 3**

Öffne die Datei Wohnortgroesse.xlsx [2], die Daten zur Größe der Wohnorte von Kindern und Jugendlichen bis 18 Jahren in Niedersachsen enthält.

a) Erstelle ein gestapeltes Balkendiagramm, in dem für jede Wohnortgröße die Anzahl der dort lebenden Jugendlichen in den Altersgruppen aufgetragen ist.

b) Füge eine Überschrift und einen Achsentitel an der vertikalen Achse ein.






Datenquellen:

[1] https://meteostat.net/de/place/de/hameln?s=D3675&t=2023-02-01/2023-02-28 (Stand April 2023)

[2] https://www1.nls.niedersachsen.de/statistik/html/default.asp; 12411 - Fortschreibung des Bevölkerungsstandes; Bevölkerung nach Altersgruppen (23) und Geschlecht (Gemeinde) (Stand Februar 2023)

[3] Flächenerhebung nach Art der tatsächlichen Nutzung, https://www.statistik.niedersachsen.de/flaechenerhebung/flachenerhebung-nach-art-der-tatsachlichen-nutzung-statistische-berichte-87671.html (Stand April 2023)

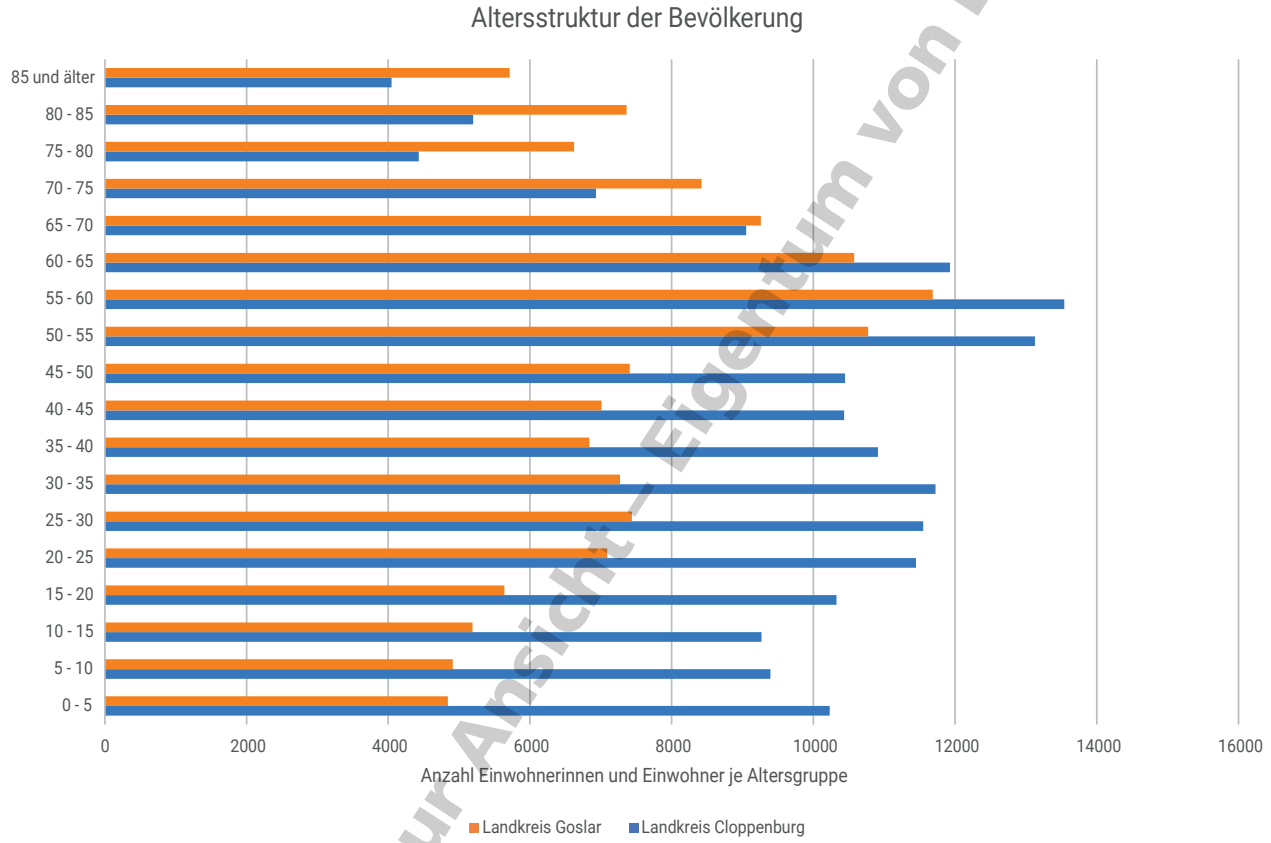
# Daten graphisch darstellen

**Aufgabe 4**

Öffne die Datei Altersstruktur_CLP_GS.xlsx [2], die Daten zur Altersstruktur der Bevölkerung der Landkreise Cloppenburg und Goslar enthält.

a) Erstelle ein Balkendiagramm, das die Altersstruktur beider Landkreise enthält.

b) Füge eine Überschrift und einen Achsentitel an der horizontalen Achse ein.






Datenquellen:

[1] https://meteostat.net/de/place/de/hameln?s=D3675&t=2023-02-01/2023-02-28 (Stand April 2023)

[2] https://www1.nls.niedersachsen.de/statistik/html/default.asp; 12411 - Fortschreibung des Bevölkerungsstandes; Bevölkerung nach Altersgruppen (23) und Geschlecht (Gemeinde) (Stand Februar 2023)

[3] Flächenerhebung nach Art der tatsächlichen Nutzung, https://www.statistik.niedersachsen.de/flaechenerhebung/flachenerhebung-nach-art-der-tatsachlichen-nutzung-statistische-berichte-87671.html (Stand April 2023)

# Sensoren und Aktoren ansteuern

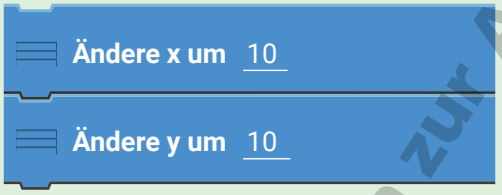
## Positionen und Bewegungen

Die gesamte Programmierung in Pocket Code erfolgt durch Bausteine. Sobald eine Figur in ein Projekt eingefügt wird, fügt die App selbst die ersten Bausteine eines Skripts ein, mit dem die Figur auf dem Bildschirm platziert wird.


Die voreingestellten Werte 100 und 200 der beiden Parameter des zweiten Bausteins sind dabei schon an die Position der Figur angepasst.


Im weiteren Verlauf eines Skripts kann man mit diesen Bausteinen eine Figur anweisen, sich zu bestimmten x- oder y-Koordinaten zu bewegen.

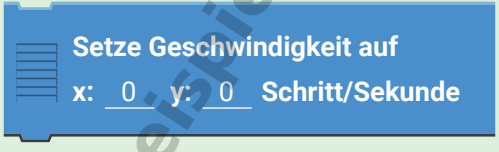
Dabei können absolute Positionen definiert werden oder relative Koordinaten mit Bezug zur aktuellen Position, indem im Formeleditor aus den Eigenschaften die aktuelle „Position x“ oder die aktuelle „Position y“ eingesetzt werden.



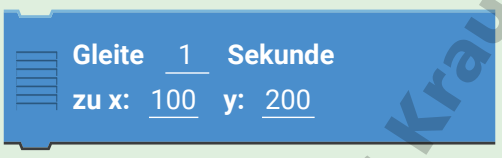

Mit diesen Bausteinen lassen sich entsprechend Figuren in x- und y-Richtung bewegen.

## Geschwindigkeit

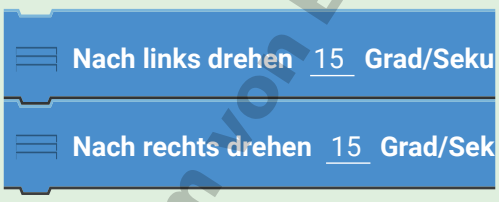
Die Geschwindigkeit, mit der sich eine Figur bewegt, wird mit diesem Baustein vorgegeben.




Alternativ kann man mit diesem Baustein vorgeben, welche Strecke eine Figur in einer bestimmten Zeit zurücklegt.




Soll sich eine Figur drehen, kann die Rotationsgeschwindigkeit mit diesen Bausteinen eingestellt werden.

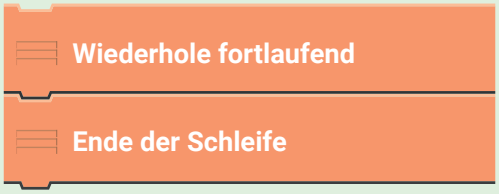



Für alle diese Bausteine gilt, dass die voreingestellten Parameter im Formel-Editor durch andere Zahlenwerte, Variablen, Funktionen oder die Messwerte von Sensoren ersetzt werden können. Dabei können unterschiedliche Größen auch durch logische Operatoren miteinander verknüpft werden.

## Wiederholungen und Abfragen

Häufig sollen bestimmte Anweisungen nicht nur einmal ausgeführt, sondern wiederholt werden.


Mit diesem Baustein wird definiert, wie oft die darunter platzierten Anweisungen wiederholt werden sollen.




Mit diesem Baustein erreicht man, dass die Anweisungen fortlaufend wiederholt werden, solange das Programm aktiv ist.

Den Block „Ende der Schleife“ fügt die Pocket Code App bei allen Wiederholungen automatisch ein. Er ist wichtig, da nur die Bausteine zwischen Start und Ende einer Wiederholungsschleife wiederholt werden.

# Sensoren und Aktoren ansteuern

## Aufgabe 1

**Hintergrund**

holz.jpg

**Figur**

kugel.png

**Story**

Die Kugel erscheint bei Programmstart in der Mitte. Wenn das Smartphone geneigt wird, soll sich die Kugel wie beim Balancieren auf einem Brett in die jeweilige Richtung bewegen. Je stärker die Neigung, desto schneller soll sich die Kugel bewegen. Sie soll dabei am Rand abprallen.

a) Erstelle ein neues Projekt „Kugel“ mit vertikaler Ausrichtung. Lade als Hintergrundbild die Datei holz.jpg.
Füge eine Figur namens „Kugel“ hinzu und wähle das Bild kugel.png aus. Platziere die Figur visuell in der Mitte des Bildschirms.

b) Öffne das Menü der Figur und korrigiere die Anfangsposition auf x = 0 / y = 0.


c) Füge aus der Kategorie „Steuerung“ den Baustein „Wiederhole fortlaufend“ ein. Schiebe ihn unter den Baustein, der die Anfangsposition setzt.

# Sensoren und Aktoren ansteuern

d) Füge aus der Kategorie „Bewegung“ den Baustein „Setze Geschwindigkeit auf“ ein. Schiebe ihn zwischen die Blöcke „Wiederhole fortlaufend“ und „Ende der Schleife“.

Ändere die Parameter für x und y im Formel-Editor. Wähle dort die Rubrik „Sensoren“ und darin „Neigung x“ für den x-Parameter bzw. „Neigung y“ für den y-Parameter.


e) Füge aus der Kategorie „Bewegung“ zwei weitere Bausteine ein: „Ändere x um“ und „Ändere y um“ und schiebe sie unter den Geschwindigkeits-Baustein.

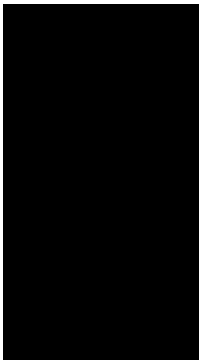
Ändere die Parameter für x und y im Formel-Editor. Wähle dazu aus der Rubrik „Sensoren“ wieder „Neigung x“ und „Neigung y“, setze in den beiden Formel aber jeweils ein Minus (-) davor.


f) Füge als letzten Baustein innerhalb der Schleife aus der Kategorie „Bewegung“ den Baustein „Pralle vom Rand ab“ ein.

# Sensoren und Aktoren ansteuern

## Aufgabe 2

**Hintergrund**



schwarz.jpg

**Figur**

wirbel.png

**Story**

Der Wirbel erscheint bei Programmstart in der Mitte.
Je nach verwendetem Sensorsignal dreht er sich umso schneller rechts herum, je

- lauter es in der Umgebung ist.
- weiter oben oder unten der Bildschirm berührt wird.

a) Erstelle ein neues Projekt „Lärm“ mit vertikaler Ausrichtung. Lade als Hintergrundbild die Datei schwarz.jpg.

b) Füge eine Figur „Wirbel“ hinzu und wähle das Bild wirbel.png aus. Lass die Figur automatisch in der Mitte des Bildschirms platzieren.


c) Füge aus der Kategorie „Ereignisse“ den Baustein „Wenn Szene startet“ ein.

d) Füge aus der Kategorie „Steuerung“ den Baustein „Wiederhole fortlaufend“ ein.


# Sensoren und Aktoren ansteuern

e) Füge aus der Kategorie „Bewegung“ den Baustein „Nach rechts drehen ... Grad/Sekunde“ ein. Setze den Parameter auf „15 x Umgebungslautstärke“.


Alternative zur Umgebungslautstärke:

- Setze den Parameter auf „Absoluter Wert(Bildschirm berührt y)“.

# Lauflängencodierung

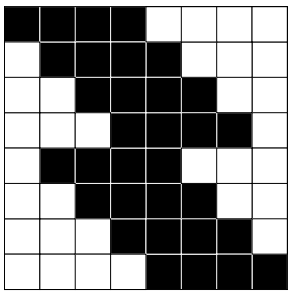


Bild 1

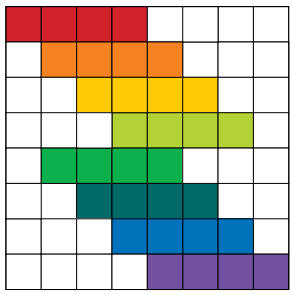


Bild 2

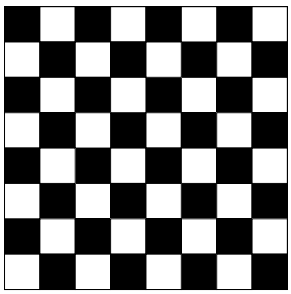


Bild 3

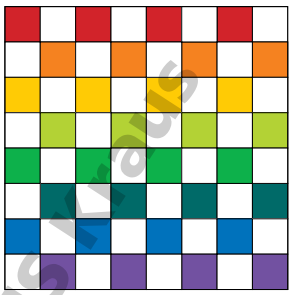


Bild 4

Digitale Bilder bestehen aus unzähligen kleinen, einfarbigen Kästchen, die Pixel genannt werden. Sie sind so klein, dass wir sie normalerweise gar nicht wahrnehmen. Eine Bilddatei enthält für jedes dieser Kästchen die entsprechende Farbinformation.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Der Inhalt von Bild 1 lässt sich ganz einfach notieren: eine 1 für jedes schwarze und eine 0 für jedes weiße Kästchen. Für jedes der 64 Kästchen wird so ein Bit Speicherplatz benötigt. Insgesamt also 64 Bit bzw. 8 Byte.

Für die farbige Variante in Bild 2 wird analog für jedes Kästchen die Farbe als RGB-Farbe notiert. Eine RGB-Farbe wird mit drei Werten für Rot, Grün und Blau zwischen 0 und 255 angegeben. Für jeden dieser Werte werden 8 Bit, für die RGB-Farbe insgesamt 3 × 8 Bit = 24 Bit Speicherplatz benötigt. Das gesamte Bild 2 ist also 64 x 24 Bit = 1 536 Bit bzw. 192 Byte groß.

Wenn wir die Bilder 1 und 2 zeilenweise „lesen", folgen stets mehrere Pixel einer Farbe aufeinander. Derartige Wiederholungen von Zeichen, Symbolen oder Zahlen finden sich häufig in Dateien. Es bietet sich also an, Folgen identischer Symbole durch deren Anzahl und das Symbol zu ersetzen.

Die Folgen identischer Symbole wurden zunächst englisch als „Run" bezeichnet. Das auf dieser Grundidee basierende Datenkompressionsverfahren erhielt den englischen Namen run-length encoding, oder kurz RLE. Davon wurden später die deutschen Bezeichnungen „Lauf" und Lauflängencodierung abgeleitet.

Das Verfahren wird technisch als Vorcodierungsschritt bei der Bildkompression (z.B. JPEG) eingesetzt.

Wenden wir die Lauflängencodierung auf unser Beispielbild 1 an, erhalten wir

| | | | | |
|---|---|---|---|---|
| 4 × 1 | 5 × 0 | 4 × 1 | 5 × 0 | 4 × 1 |
| 5 × 0 | 4 × 1 | 2 × 0 | 4 × 1 | 5 × 0 |
| 4 × 1 | 5 × 0 | 4 × 1 | 5 × 0 | 4 × 1 |

bzw. in Binärschreibweise

| | | | | |
|---|---|---|---|---|
| 100 1 | 101 0 | 100 1 | 101 0 | 100 1 |
| 101 0 | 100 1 | 010 0 | 100 1 | 101 0 |
| 100 1 | 101 0 | 100 1 | 101 0 | 100 1 |

Für die Codierung jedes einzelnen der 15 Läufe werden 3 Bit + 1 Bit benötigt. Das gesamte Bild benötigt folglich 15 × 4 Bit = 60 Bit Speicherplatz.

Wie stark ein Bild im Vergleich zur Originalgröße verkleinert wurde, gibt die Kompressionsrate an. Sie wird mit dieser Formel berechnet:

$$\text{Kompressionsrate} = 100 \times \left(1 - \frac{\text{komprimiert}}{\text{Originalgröße}}\right) \%$$

$$\text{Kompressionsrate} = 100 \times \left(1 - \frac{60 \text{ Bit}}{64 \text{ Bit}}\right) = 6{,}25 \%$$

Für die Codierung der 15 Läufe im farbigen Bild 2 werden jeweils 3 Bit für die Kästchenzahl und 24 Bit für die Farbe benötigt. Insgesamt ergibt das 15 × 27 Bit = 405 Bit. Daraus ergibt sich eine Kompressionsrate von 73,6 %. Diese hohe Kompressionsrate resultiert vor allem daraus, dass pro Lauf die Farbe nur einmal codiert werden muss.

Die Lauflängencodierung führt jedoch nicht in jedem Fall zu einem geringeren Speicherbedarf. In Dateien, die nur wenige Wiederholungen enthalten, erhält man auch bei farbigen Bildern nur eine sehr geringe Kompressionsrate oder auch einen leichten Größenzuwachs (z. B. auf 1 560 Bit bei Bild 4).

In ungünstigen Fällen wird die Datei durch die Lauflängencodierung sogar deutlich größer; im Bild 3 beispielsweise mehr als doppelt so groß.

# Lauflängencodierung

**Aufgabe 1**
Beschreibe, welche Grundidee der Lauflängencodierung zugrunde liegt und wie man bei der Codierung vorgeht.

Bei der Lauflängencodierung wird ausgenutzt, dass in Dateien häufig mehrere identische Pixel, Zeichen, Symbole oder Zahlen aufeinanderfolgen.

Bei der Codierung werden diese Folgen identischer Symbole durch deren Anzahl und das Symbol ersetzt.

**Aufgabe 2**
Für welche Art Bilder liefert das Verfahren der Lauflängencodierung hohe Kompressionsraten?

Die Lauflängencodierung liefert bei farbigen Bildern mit wenigen Farbwechseln die größten Kompressionsraten.

**Aufgabe 3**
Für welche Art Bilder sind die mit der Lauflängencodierung komprimierten Bilder größer als das Original?

Schwarz-weiß-Bilder mit vielen Wechseln zwischen schwarz und weiß und farbige Bilder mit wenigen aufeinanderfolgenden identisch gefärbten Pixeln sind mit der Lauflängencodierung meist größer als das Original.

**Aufgabe 4**
Das sind die Daten eines acht Pixel breiten Bildes:

11110000011110000011110000111100
00111100001111000001111000001111

a) Zeichne das Bild.

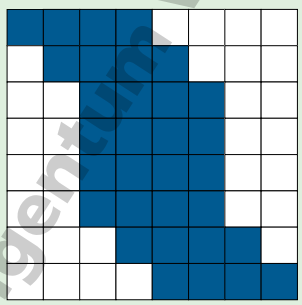
b) Berechne die Datenmenge der Bilddaten.

c) Codiere die Bilddaten mit der Lauflängencodierung.

d) Berechne die Datenmenge der komprimierten Bilddaten.

e) Wie groß ist die Kompressionsrate?

a)



b) 64 x 1 Bit = 64 Bit

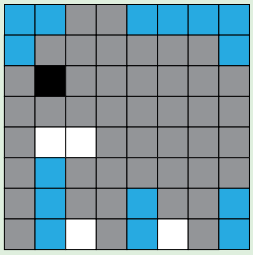
c) 4 x 1, 5 x 0, 4 x 1, 5 x 0,
4 x 1, 4 x 0, 4 x 1, 4 x 0,
4 x 1, 4 x 0, 4 x 1, 5 x 0,
4 x 1, 5 x 0, 4 x 1

d) 15 Läufe, maximal 5 lang, 3 Bit, Farbe 1 Bit
15 x (3 + 1) Bit = 60 Bit

e) Kompressionsrate 6,25 %

# Lauflängencodierung

## Aufgabe 5



a) Berechne die Datenmenge des Bildes.

b) Wie groß ist die Datenmenge, nachdem das Bild mit der Lauflängencodierung komprimiert wurde?
Beachte, wie viel Bit du für die Speicherung der maximalen Kästenchanzahl der Läufe benötigst.

c) Berechne die erzielte Kompressionsrate.

a) 64 Kästchen x 24 Bit = 1536 Bit

b) 2 x blau, 2 x grau, 5 x blau, 6 x grau,
1 x blau, 1 x grau, 1 x schwarz, 15 x grau,
2 x weiß, 6 x grau, 1 x blau, 7 x grau,
1 x blau, 2 x grau, 1 x blau, 2 x grau,
1 x blau, 1 x grau, 1 x blau, 1 x weiß,
1 x grau, 1 x blau, 1 x weiß, 1 x grau, 1 x blau

25 Läufe, maximal 15 lang, 4 Bit,
Farbe 24 Bit
25 x (4 + 24) Bit = 700 Bit

c) Kompressionsrate: 54,4 %

## Aufgabe 6

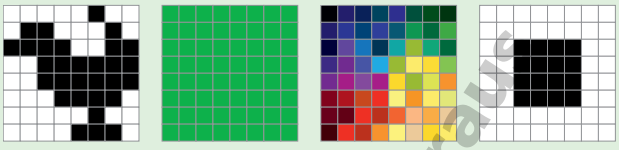


Bild 1 Bild 2 Bild 3 Bild 4

a) Schätze ein, ob die Lauflängencodierung bei den vier Bildern zu einer Verringerung der Datenmenge führen wird?

b) Berechne die Datenmenge der vier Bilder.

c) Wie groß ist die Datenmenge der vier Bilder, nachdem sie mit der Lauflängencodierung komprimiert wurden?

d) Berechne die erzielten Kompressionsraten.
War deine Einschätzung aus Aufgabe a) richtig?

Bild 1

b) 64 x 1 Bit = 64 Bit

c) 00000100
01100010
11110011
00111111
00111111
00011110
00000100
00001110

5 x 0, 1 x 1, 3 x 0, 2 x 1, 3 x 0, 1 x 1, 1 x 0,
4 x 1, 2 x 0, 2 x 1, 2 x 0, 6 x 1, 2 x 0, 6 x 1,
3 x 0, 4 x 1, 6 x 0, 1 x 1, 6 x 0, 3 x 1, 1 x 0

21 Läufe, maximal 6 lang, 3 Bit, Farbe 1 Bit
21 x (1 + 3) Bit = 84 Bit

d) das „komprimierte" Bild ist größer

# Lauflängencodierung

## Bild 2

b) 64 Kästchen x 24 Bit = 1536 Bit

c) 1 Lauf, 64 lang, 7 Bit, Farbe 24 Bit
1 x (7 + 24) Bit = 31 Bit

d) Kompressionrate: 98 %

## Bild 3

b) 64 Kästchen x 24 Bit = 1536 Bit

c) 64 Läufe, 1 lang, 1 Bit, Farbe 24 Bit
64 x (1 + 24 Bit) = 1600 Bit

d) das „komprimierte" Bild ist größer

## Bild 4

b) 64 x 1 Bit = 64 Bit

c) 00000000
00000000
00111100
00111100
00111100
00111100
00000000
00000000

18 x 0, 4 x 1, 4 x 0, 4 x 1,
4 x 0, 4 x 1, 4 x 0, 4 x 1, 18 x 0

9 Läufe, maximal 18 lang, 5 Bit, Farbe 1 Bit
9 x (5 + 1) Bit = 54 Bit

d) Kompressionsrate: 15,6 %

# Auflösung und Farbtiefe

**Bild 1**
Auflösung 300 ppi / 300 dpi
Farbtiefe 16 Bit
Dateigröße 1,39 Mebibyte

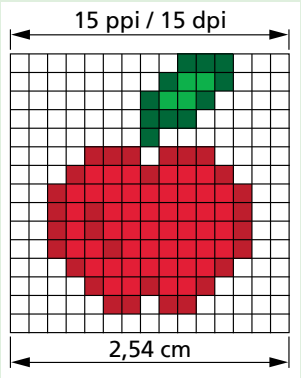
**Bild 2**
Auflösung 300 ppi / 300 dpi
Farbtiefe 8 Bit
Dateigröße 731 Kibibyte

**Bild 3**
Auflösung 72 ppi / 72 dpi
Farbtiefe 16 Bit
Dateigröße 106 Kibibyte

**Bild 4**
Auflösung 72 ppi / 72 dpi
Farbtiefe 8 Bit
Dateigröße 65,5 Kibibyte

## Auflösung

Digitale Bilder bestehen aus kleinen, einfarbigen Kästchen, den Pixeln. Für die Qualität eines Bildes ist es unter anderem wichtig, wie dicht die Pixel angeordnet sind. Befinden sich viele Pixel auf engem Raum, spricht man von einer hohen Auflösung.




Die Auflösung eines Bildes wird in dpi oder ppi angegeben. dpi wird für Druckdaten verwendet und bedeutet Dots per Inch, also Punkte pro Inch. Inch ist eine Maßeinheit des angloamerikanischen Maßsystems und entspricht 2,54 cm.

Die in den Bildern 1 und 2 verwendete Auflösung von 300 dpi entspricht der üblichen Bildauflösung von Bildern für den Druck.

Bei Bildern für digitale Medien wird die Auflösung eher in ppi angegeben. Das bedeutet Pixel pro Inch.

## Farbtiefe

Die drei Farbkanäle Rot, Grün und Blau in farbigen Bildern nach dem RGB-Farbschema können in gewöhnlichen Bildern Werte zwischen 0 und 255 annehmen. Damit stehen pro Farbkanal $2^8$ Abstufungen zur Verfügung. Daraus ergeben sich $(2^8)^3$, also über 16 Millionen unterschiedliche Farben.

Für die Anzahl der verfügbaren Farben für ein Pixel wird der Begriff Farbtiefe verwendet.

Genau genommen bezeichnet der Begriff aber nicht die Anzahl der Farben, sondern den Speicherplatz, der nötig ist, um die Farbinformation dieser Farben zu speichern. Für $2^8$ Abstufungen pro Farbkanal sind das 8 Bit und damit beträgt auch die Farbtiefe 8 Bit.

## Dateigröße

Die Größe einer Bilddatei entspricht der Summe des Speicherbedarfs der Farbinformationen aller Pixel und ein paar weiteren Informationen zu Format, Abmessung und Name der Datei. Die Dateigröße lässt sich daher mit dieser Formel näherungsweise berechnen:

Dateigröße ≈ Pixelanzahl × Farbtiefe × Farbkanäle

Datei Apfel ≈ 225 Pixel × 8 Bit × 3 ≈ 5 400 Bit

Da aus einer höheren Auflösung meist eine größere Pixelanzahl resultiert, haben sowohl die Auflösung als auch die Farbtiefe einen direkten Einfluss auf die Größe einer Bilddatei. Anhand der Bilder oben ist leicht zu sehen, dass eine Verdoppelung der Farbtiefe die Dateigröße annähernd verdoppelt. Im Gegenzug bewirkt eine geringere Auflösung – also weniger Pixel bei gleicher Bildbreite – eine drastische Reduktion der Dateigröße.

Möchte man die Größe eines Bildes reduzieren, beispielsweise, um es für eine Website zu verwenden, kann man eine Farbtiefe von 8 Bit anstelle von beispielsweise 16 Bit verwenden und die Auflösung verringern. Während die geringere Farbtiefe dabei vermutlich kaum auffällt, verändert die verlustbehaftete Verringerung der Pixelanzahl bei gleichbleibender Bildabmessung das Aussehen des Bildes möglicherweise deutlich.

Foto Burg Eltz: Walter Kärcher (Pixabay)

# Auflösung und Farbtiefe

**Aufgabe 1**
Ein Bild ist 22 × 22 cm groß und enthält 1 299 × 1 299 Pixel.

Wie groß ist die Auflösung?

Umrechnung Seitenlänge in die Einheit Inch:
22 cm : 2,54 = 8,66 Inch

Pixelanzahl durch Seitenlänge in Inch teilen:
1299 Pixel : 8,66 Inch = 150 ppi

Die Auflösung beträgt 150 ppi bzw. 150 dpi.

**Aufgabe 2**
Wie viele Farben stehen bei einem RGB-Bild für jedes einzelne Pixel zur Verfügung?

a) bei 8 Bit Farbtiefe

b) bei 16 Bit Farbtiefe

c) bei 4 Bit Farbtiefe

| | Farbtiefe | Farben |
|---|---|---|
| a) | 8 Bit | $(2^8)^3 = 16\,777\,216$ |
| b) | 16 Bit | $(2^{16})^3 = 281\,474\,976\,710\,656$ |
| c) | 4 Bit | $(2^4)^3 = 4\,096$ |

**Aufgabe 3**
Berechne die ungefähre Dateigröße für ein RGB-Bild mit 2 500 × 2 000 Pixeln und 16 Bit Farbtiefe.

Dateigröße = 2500 x 2000 x 16 Bit x 3

Dateigröße = 240 000 000 Bit
= 30 000 000 Byte
= 29 296,875 Kibibyte
= 28,61 Mebibyte

**Aufgabe 4**
Auf wie viel Prozent kann die Größe einer RGB-Bilddatei ungefähr reduziert werden, wenn sie in ein Graustufenbild umgewandelt wird?

Hinweis: Überleg zuerst, wie viele Farbkanäle ein Graustufenbild hat.

Die Dateigröße wird mit dieser Formel berechnet:
Dateigröße = Pixelanzahl x Farbtiefe x Anzahl Farbkanäle

Graustufenbilder haben nur einen Farbkanal für Schwarz mit der entsprechenden Farbtiefe des Bildes, z.B. 8 Bit, also 256 Abstufungen von schwarz bis weiß.

Dateigröße RGB-Farben = Pixelanzahl x Farbtiefe x 3
Dateigröße Graustufen = Pixelanzahl x Farbtiefe x 1

Die Dateigröße eines RGB-Bildes kann durch Umwandeln in ein Graustufenbild auf etwa ein Drittel reduziert werden.

# Auflösung und Farbtiefe

**Aufgabe 5**

Eine Digitalkamera hat einen Bildsensor mit 20 Megapixeln und arbeitet mit 8 Bit Farbtiefe und Standard-RGB-Farben.

Wieviel Speicherplatz benötigt jedes Bild ungefähr, das mit der Kamera aufgenommen wird?

20 Megapixel = 20 Millionen Pixel

Speicherplatz = 20 000 000 x 8 x 3
= 480 000 000 Bit
= 60 000 000 Byte
= 60 000 Kilobyte (58 593,75 Kibibyte)
= 60 Megabyte (57,22 Mebibyte)

**Aufgabe 6**

Du hast ein RGB-Bild mit 8 Bit Farbtiefe vorliegen. Es ist 5 000 × 3 300 Pixel groß.

Du möchtest es mit einer Breite von 20 cm und einer Auflösung von 300 dpi drucken.

Die Bilddatei soll dafür so klein wie möglich sein.

Um wie viel Prozent kannst du die Dateigröße vor dem Druck reduzieren?

Größe Ausgangsdatei:
5 000 x 3 300 x 8 x 3 = 396 000 000 Bit

Endmaß in Inch:
Breite 20 cm = 7,874 Inch
Breite 7,874 Inch x 300 dpi = 2362 Pixel

Berechnung der Bildhöhe:

$$\frac{2\,362 \text{ Pixel}}{5\,000 \text{ Pixel}} = \frac{\text{Bildhöhe } x}{3\,300 \text{ Pixel}}$$

$$\text{Bildhöhe } x = \frac{2\,362 \times 3\,300}{5\,000} = 1\,559 \text{ Pixel}$$

2 362 x 1 559 x 8 x 3 = 88 376 592 Bit

$$100 \times \left(1 - \frac{88\,376\,592 \text{ Bit}}{396\,000\,000 \text{ Bit}}\right) = 100 \times (1 - 0{,}22317321) = 77{,}7\,\%$$

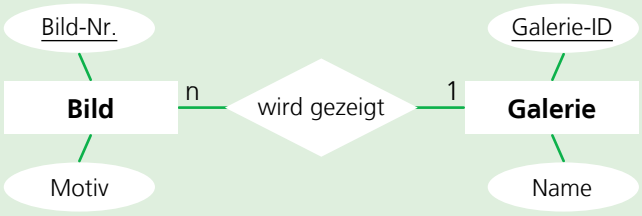
# Relationales Datenbankschema

Das relationale Datenbankschema ist ein häufig eingesetztes Datenbankschema, das auch von den populärsten Datenbankmanagementsystemen Oracle und MySQL verwendet wird. [1)]

Das Wort Relation geht auf das lateinische relatio zurück, das u.a. für Beziehung oder Verhältnis steht. Ein relationales Datenbankschema beruht auf Tabellen mit Eigenschaften, die zueinander in Beziehung stehen.

### Vom ER-Diagramm zum Datenbankschema

Zur Veranschaulichung greifen wir wieder auf unser Beispiel aus der vorherigen Lektion zurück. Dieses einfache ER-Diagramm besteht aus zwei Entitätstypen mit jeweils einer Eigenschaft (Motiv, Name), die durch eine 1:n-Beziehung miteinander verbunden sind.




Im Datenbankschema wird für jeden Entitätstyp eine Tabelle erstellt. Die Tabellen haben jeweils eine Spalte für die Eigenschaften Motiv bzw. Name. Zusätzlich werden eine Spalte mit einer Bild-Nummer und eine Spalte mit einer Galerie-ID eingefügt. Sie dienen als so genannter Primärschlüssel und ermöglichen eine eindeutige Unterscheidung zweier Bilder auch dann, wenn die Motive gleich benannt sind. Im ER-Diagramm werden die Primärschlüssel durch Unterstreichen gekennzeichnet.

**Bild**

| Bild-Nr. | Motiv |
|---|---|
| 023 | Rafting |
| 286 | Mondlicht |
| 541 | Skyline |
| 881 | Alte Brücke |

**Galerie**

| Galerie-ID | Name |
|---|---|
| gal_14 | Galerie 14 |
| kun_hof | Kunst im Hof |
| par_gal | Park-Galerie |

Die Beziehung zwischen den beiden Tabellen wird hergestellt, indem der Primärschlüssel einer Tabelle als so genannter Fremdschlüssel in die andere Tabelle aufgenommen wird. Dafür haben wir in unserem Beispiel theoretisch zwei Möglichkeiten: die Bild-Nr. wird in die Tabelle Galerie eingefügt oder die Galerie-ID wird in die Tabelle Bild eingefügt. Beginnen wir mit der ersten Möglichkeit.

Wenn in einer Galerie mehrere Bilder gezeigt werden, führt das Eintragen der Bild-Nummer in die Tabelle Galerie zwangsläufig zu einer Verdopplung einzelner Zeilen. Damit ist zum einen der Primärschlüssel „Galerie-ID“ nicht mehr eindeutig und zum anderen wird der Name der Galerien wiederholt. Eine solche Wiederholung nennt man auch Redundanz.

**Bild**

| Bild-Nr. | Motiv |
|---|---|
| 023 | Rafting |
| 286 | Mondlicht |
| 541 | Skyline |
| 881 | Alte Brücke |

**Galerie**

| Galerie-ID | Name | Bild-Nr. |
|---|---|---|
| gal_14 | Galerie 14 | 541 |
| gal_14 | Galerie 14 | 881 |
| kun_hof | Kunst im Hof | 286 |
| par_gal | Park-Galerie | 023 |

Redundanzen in Datenbanken sind unerwünscht, weil sie die Konsistenz der Daten gefährden, also nicht mehr sichergestellt ist, dass die Daten korrekt, einheitlich und aktuell sind. Wenn beispielsweise in einer Zeile die „Galerie 14“ in „Galerie 144“ umbenannt würde, wäre nicht mehr klar, wie die Galerie tatsächlich heißt.

Der Fremdschlüssel wird daher bei einer **1:n-Beziehung** stets in die Tabelle des Entitätstyps eingefügt, bei dem im ER-Diagramm das n steht. Nur dann führt das Einfügen nicht zu einer unzulässigen Verdoppelung einzelner Primärschlüssel.

**Bild**

| Bild-Nr. | Motiv | Galerie-ID |
|---|---|---|
| 023 | Rafting | par_gal |
| 286 | Mondlicht | kun_hof |
| 541 | Skyline | gal_14 |
| 881 | Alte Brücke | gal_14 |

**Galerie**

| Galerie-ID | Name |
|---|---|
| gal_14 | Galerie 14 |
| kun_hof | Kunst im Hof |
| par_gal | Park-Galerie |

Für **n:m-Beziehungen** würde das Einfügen der Fremdschlüssel zwangsläufig zu Redundanzen in den Tabellen führen. Deshalb erstellt man hier noch eine weitere Tabelle, die nur die Beziehung zwischen den beiden Primärschlüsseln herstellt.

**Künstler**

| Künstler-ID | Name |
|---|---|
| meye | Meyer |
| muel | Müller |
| schm | Schmidt |

**Galerie**

| Galerie-ID | Name |
|---|---|
| gal_14 | Galerie 14 |
| kun_hof | Kunst im Hof |
| par_gal | Park-Galerie |

**stellt aus**

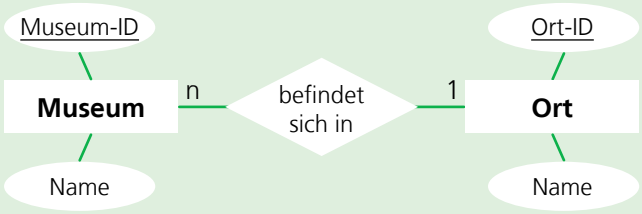
| Künstler-ID | Galerie-ID |
|---|---|
| meye | gal_14 |
| meye | kun_hof |
| muel | gal_14 |
| muel | kun_hof |
| muel | par_gal |
| schm | gal_14 |
| schm | par_gal |

[1)] https://db-engines.com/de/ranking (Stand Juli 2024)

# Relationales Datenbankschema

**Aufgabe 1**

Erstelle für das folgende ER-Diagramm alle für ein relationales Datenbankschema benötigten Tabellen mit den erforderlichen Primärschlüsseln und Fremdschlüsseln.




Befülle die Tabellen mit diesen Beispieldaten:

- Schloss Benrath in Düsseldorf
- Deutsches Fußballmuseum in Dortmund
- Museum Folkwang in Essen
- Museum Ludwig in Köln
- Schokoladenmuseum in Köln
- Zeche Zollverein in Essen

Beispiellösung

**Museum**

| Museum-ID | Name | Ort-ID |
|---|---|---|
| benr | Schloss Benrath | duess |
| defu | Deutsches Fußballmuseum | dortm |
| folk | Museum Folkwang | essen |
| ludw | Museum Ludwig | koeln |
| scho | Schokoladenmuseum | koeln |
| zoll | Zeche Zollverein | essen |

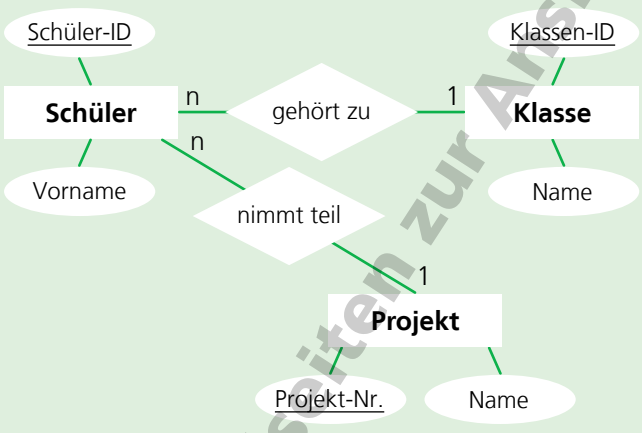
**Ort**

| Ort-ID | Name |
|---|---|
| dortm | Dortmund |
| duess | Düsseldorf |
| essen | Essen |
| koeln | Köln |

**Aufgabe 2**

Erstelle für das folgende ER-Diagramm alle für ein relationales Datenbankschema benötigten Tabellen mit den erforderlichen Primärschlüsseln und Fremdschlüsseln.

Befülle die Tabellen mit eigenen Beispieldaten.




Beispiellösung

**Schüler**

| Schüler-ID | Vorname | Klassen-ID | Projekt-ID |
|---|---|---|---|
| 1003 | Linus | 8a | 24D |
| 1213 | Murat | 8b | 24C |
| 1345 | Nele | 8c | 24A |
| 1387 | Milan | 8c | 24B |
| 1444 | Marie | 8b | 24C |
| 1502 | Özlem | 8a | 24A |

**Klasse**

| Klassen-ID | Klasse |
|---|---|
| 8a | Klasse 8 a |
| 8b | Klasse 8 b |
| 8c | Klasse 8 c |

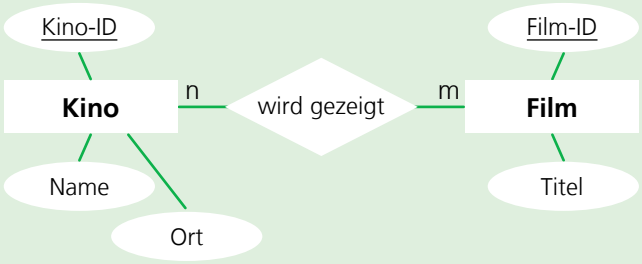
**Projekt**

| Projekt-ID | Name |
|---|---|
| 24A | Gebärdensprache lernen |
| 24B | Müll sammeln am See |
| 24C | Physik im Kindergarten |
| 24D | Bauprojekt im Tierheim |

# Relationales Datenbankschema

**Aufgabe 3**

Erstelle für das folgende ER-Diagramm alle für ein relationales Datenbankschema benötigten Tabellen mit den erforderlichen Primärschlüsseln und Fremdschlüsseln.




Befülle die Tabellen mit diesen Beispieldaten:

**Filme**

- Ich war neunzehn
- Die Legende von Paul und Paula
- Das Boot
- Der Himmel über Berlin
- Gegen die Wand

**Kinos**

- Cinema, Düsseldorf
- Camera, Dortmund
- filmforum, Duisburg
- Lichtburg, Essen
- Metropolis, Köln

## Beispiellösung

### Kino

| Kino-ID | Name | Ort |
|---|---|---|
| d-cin | Cinema | Düsseldorf |
| do-cam | Camera | Dortmund |
| du-fil | filmforum | Duisburg |
| e-lic | Lichtburg | Essen |
| k-met | Metropolis | Köln |

### Film

| Film-ID | Titel |
|---|---|
| D097 | Ich war neunzehn |
| D143 | Die Legende von Paul und Paula |
| D384 | Das Boot |
| D420 | Der Himmel über Berlin |
| D523 | Gegen die Wand |

### wird gezeigt

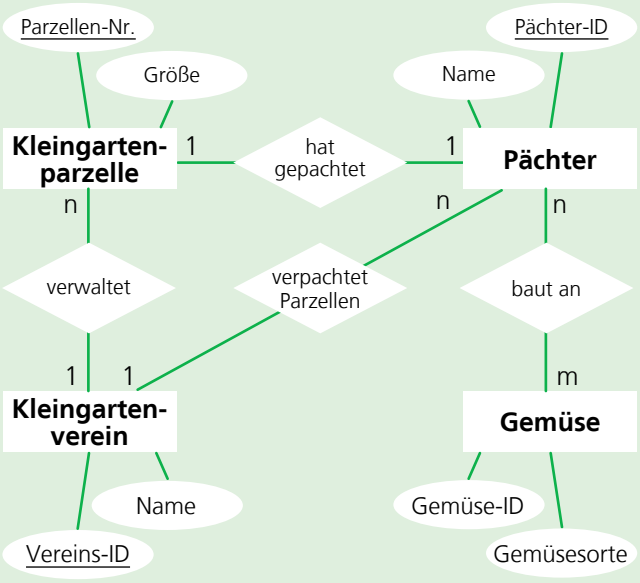
| Kino-ID | Film-ID |
|---|---|
| d-cin | D143 |
| d-cin | D523 |
| do-cam | D384 |
| do-cam | D420 |
| du-fil | D097 |
| du-fil | D143 |

| Kino-ID | Film-ID |
|---|---|
| du-fil | D523 |
| e-lic | D143 |
| e-lic | D384 |
| k-met | D097 |
| k-met | D384 |
| k-met | D420 |

# Relationales Datenbankschema

**Aufgabe 4**

Erstelle für das folgende ER-Diagramm alle für ein relationales Datenbankschema benötigten Tabellen mit den erforderlichen Primärschlüsseln und Fremdschlüsseln.




Befülle die Tabellen mit diesen Beispieldaten:

**Kleingartenvereine**
- Am Mühlenberg
- Naturfreunde

**Pächter**
- Schubert
- Kowalski
- Yılmaz
- Otto

**Gemüsesorte**
- Blumenkohl
- Buschbohnen
- Feldsalat
- Kohlrabi
- Radieschen
- Zucchini

**Kleingartenparzelle**
- 370 m²
- 425 m²
- 300 m²
- 390 m²

## Beispiellösung

### Kleingartenparzelle

| Parzellen-Nr. | Größe | Vereins-ID | Pächter-ID |
|---|---|---|---|
| mueh-025 | 370 m² | mueh | 1785 |
| mueh-103 | 425 m² | mueh | 366 |
| natu-001 | 300 m² | natu | 3389 |
| natu-251 | 390 m² | natu | 2417 |

### Pächter

| Pächter-ID | Name | Vereins-ID |
|---|---|---|
| 366 | Schubert | mueh |
| 1785 | Kowalski | mueh |
| 2417 | Yilmaz | natu |
| 3389 | Otto | natu |

### Gemüse

| Gemüse-ID | Gemüsesorte |
|---|---|
| blukoh | Blumenkohl |
| buschb | Buschbohnen |
| feldsa | Feldsalat |
| kohlra | Kohlrabi |
| radies | Radieschen |
| zucchi | Zucchini |

### Kleingartenverein

| Vereins-ID | Name |
|---|---|
| mueh | Am Mühlenberg |
| natu | Naturfreunde |

### baut an

| Pächter-ID | Gemüse-ID |
|---|---|
| 366 | buschb |
| 366 | feldsa |
| 366 | kohlra |
| 366 | radies |
| 1785 | blukoh |
| 1785 | feldsa |
| 1785 | zucchi |
| 2417 | buschb |
| 2417 | kohlra |
| 2417 | zucchi |
| 3389 | blukoh |
| 3389 | radies |

# Daten abfragen mit SQL

Für SQL wird häufig die nicht ganz korrekte Bezeichnung „Structured Query Language“ (deutsch: Strukturierte Abfrage-Sprache) verwendet. Das deutet darauf hin, dass das Abfragen von Informationen aus Datenbanken die vordringliche Anwendung für diese Sprache war und ist.

Anhand der beiden folgenden Tabellen zeigen wir die wichtigsten dafür benötigten SQL-Befehle.

**Schiffe** [1)]

| schiff_id | name | container | reeder_id |
|---|---|---|---|
| 1 | Emma Maersk | 17816 | 2 |
| 2 | Ever Alot | 24000 | 3 |
| 3 | Madrid Maersk | 20568 | 2 |
| 4 | MSC Irina | 24346 | 1 |
| 5 | MSC Jade | 19437 | 1 |

**Reedereien**

| reeder_id | name | sitz |
|---|---|---|
| 1 | A. P. Moller-Maersk Group | Kopenhagen |
| 2 | Evergreen Marine Corp. (Taiwan) Ltd. | Taipeh |
| 3 | Mediterranean Shipping Company | Genf |

Der wichtigste Befehl für SQL-Abfragen lautet `SELECT ... FROM`. Wird er mit einem Sternchen verwendet, liefert die Anweisung den gesamten Inhalt der entsprechenden Tabelle. Werden Spaltennamen in die Anweisung eingefügt, erhält man den Inhalt der genannten Spalten.

```
SELECT * FROM Schiffe;
```

```
SELECT name, container FROM Schiffe;
```

Auch bei Abfragen kann man Bedingungen mit `WHERE` verwenden, um das Ergebnis einzuschränken. Dabei können die Vergleichsoperatoren `<`, `>`, `=` oder der `BETWEEN`-Operator eingesetzt werden.

```
SELECT name, container FROM Schiffe
WHERE container < 20000;
```

```
SELECT name, container FROM Schiffe
WHERE container BETWEEN 20000 AND 25000
ORDER BY container ASC;
```

Mit dem SQL-Befehl `ORDER BY` bewirkt man das Sortieren des Abfrageergebnisses. `ASC` führt zu einer aufsteigenden (a–z), `DESC` zu einer absteigenden (z–a) Sortierung.

Relationale Datenbanken können aus zahlreichen Tabellen bestehen, die miteinander in Beziehung stehen. Das gelingt, indem der Primärschlüssel einer Tabelle als Fremdschlüssel in eine andere Tabelle aufgenommen wird. In unseren Beispieltabellen ist die reeder_id dieser Fremdschlüssel, der die Tabellen Reedereien und Schiffe verknüpft.

Durch diese Beziehung ist es möglich, Daten aus mehreren Tabellen abzufragen.

```
SELECT Schiffe.name, Reedereien.name
FROM Schiffe
INNER JOIN Reedereien
ON Schiffe.reeder_id=Reedereien.reeder_id;
```

Sobald Daten aus mehreren Tabellen abgefragt werden, verwendet man die Spaltennamen mit vorangestelltem Tabellennamen, um Verwechslungen zu vermeiden (z. B. `Reedereien.name`).

Mit dem SQL-Befehl `INNER JOIN` wird angegeben, welche zweite Tabelle in die Abfrage aus unserer Tabelle Schiffe eingebunden werden soll. Nach dem Schlüsselwort `ON` stehen die Spaltennamen aus beiden Tabellen, die die Beziehung bilden. Die obige Anweisung liefert eine Tabelle mit den fünf Schiffen und den dazugehörenden Reedereien.

Durch das Anfügen von Bedingungen (`WHERE`) und den Befehl `ORDER BY` können auch diese Ergebnisse eingeschränkt und sortiert werden.

Der SQL-Befehl `COUNT` ermöglicht das Zählen von Zeilen in einer Tabelle, die einem Kriterium aus einer anderen Tabelle entsprechen. In unserem Beispiel werden die Schiffe gezählt, die zu einer Reederei gehören, und unter dem Spaltennamen anzahl_schiffe als Ergebnis ausgegeben.

```
SELECT Reedereien.name,
COUNT(Schiffe.reeder_id) AS anzahl_schiffe
FROM Schiffe
INNER JOIN Reedereien
ON Schiffe.reeder_id=Reedereien.reeder_id
GROUP BY Reedereien.name;
```

Das Zuordnen zu den jeweiligen Namen der Reedereien erfolgt mit Hilfe des SQL-Befehls `GROUP BY`.



[1)] https://de.wikipedia.org/wiki/Containerschiff (Stand 09/2024)

# Daten abfragen mit SQL

Für die Bearbeitung der folgenden Aufgaben werden die befüllten Tabellen Bundesländer und Großstädte benötigt. Beide Tabellen müssen zunächst per SQL-Anweisungen im Programiz-Editor angelegt und befüllt werden.

Kopiere dafür nacheinander den Inhalt der Dateien SQL-Skript_Tabelle_bundeslaender.txt und SQL-Skript_Tabelle_grossstaedte.txt in den Editor und führe sie aus.

## Aufgabe 1

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die mehr als 600000 Einwohner haben. Name und Einwohnerzahl der Städte sollen aufgelistet werden, absteigend nach der Einwohnerzahl sortiert.

Schreibe eine entsprechende SQL-Abfrage.

### Beispiellösung

```
SELECT name, einwohner
FROM Grossstaedte
WHERE einwohner > 600000
ORDER BY einwohner DESC;
```

**Output**

| name | einwohner |
|---|---|
| Berlin | 3782202 |
| Hamburg | 1910160 |
| München | 1510378 |
| Köln | 1087353 |
| Frankfurt am Main | 775790 |
| Stuttgart | 633484 |
| Düsseldorf | 631217 |
| Leipzig | 619879 |

## Aufgabe 2

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die weniger als 75 km² Fläche einnehmen. Name und Fläche der Städte sollen aufgelistet werden, aufsteigend nach der Fläche sortiert.

Schreibe eine entsprechende SQL-Abfrage.

### Beispiellösung

```
SELECT name, flaeche
FROM Grossstaedte
WHERE flaeche < 75
ORDER BY flaeche ASC;
```

**Output**

| name | flaeche |
|---|---|
| Offenbach am Main | 44.88 |
| Herne | 51.42 |
| Fürth | 63.35 |
| Recklinghausen | 66.5 |
| Moers | 67.64 |
| Remscheid | 74.52 |

# Daten abfragen mit SQL

**Aufgabe 3**
Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die weniger als 110000 Einwohner haben. Name, Einwohner und Bundesland sollen aufgelistet werden, aufsteigend nach der Einwohnerzahl sortiert.

Schreibe eine entsprechende SQL-Abfrage. Nutze die land_id und den Befehl `INNER JOIN`.

## Beispiellösung

```
SELECT Grossstaedte.name, Grossstaedte.einwohner, Bundeslaender.land
FROM Grossstaedte
INNER JOIN Bundeslaender ON Grossstaedte.land_id = Bundeslaender.land_id
WHERE Grossstaedte.einwohner < 110000
ORDER BY Grossstaedte.einwohner ASC;
```

**Output**

| name | einwohner | land |
|---|---|---|
| Cottbus/Chósebuz | 100010 | Brandenburg |
| Kaiserslautern | 101486 | Rheinland-Pfalz |
| Siegen | 102114 | Nordrhein-Westfalen |
| Hildesheim | 102325 | Niedersachsen |
| Gütersloh | 102464 | Nordrhein-Westfalen |
| Hanau | 103184 | Hessen |
| Salzgitter | 105039 | Niedersachsen |
| Moers | 105606 | Nordrhein-Westfalen |

# Daten abfragen mit SQL

**Aufgabe 4**

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die im Bundesland Nordrhein-Westfalen liegen. Name, Einwohner und Fläche sollen aufgelistet werden, aufsteigend nach dem Namen der Großstadt sortiert.

Schreibe eine entsprechende SQL-Abfrage. Nutze die land_id und den Befehl `INNER JOIN`.

## Beispiellösung

```
SELECT Grossstaedte.name, Grossstaedte.einwohner, Grossstaedte.flaeche
FROM Grossstaedte
INNER JOIN Bundeslaender ON Grossstaedte.land_id = Bundeslaender.land_id
WHERE Bundeslaender.land = "Nordrhein-Westfalen"
ORDER BY Grossstaedte.name;
```

**Output**

| name | einwohner | flaeche |
|---|---|---|
| Aachen | 252769 | 160.85 |
| Bergisch Gladbach | 112660 | 83.09 |
| Bielefeld | 338410 | 258.83 |
| Bochum | 366385 | 145.66 |
| Bonn | 335789 | 141.06 |
| Bottrop | 118705 | 100.61 |
| Dortmund | 595471 | 280.71 |
| Duisburg | 503707 | 232.84 |
| Düsseldorf | 631217 | 217.41 |
| Essen | 586608 | 210.34 |
| Gelsenkirchen | 265885 | 104.94 |
| Gütersloh | 102464 | 112.02 |
| Hagen | 190490 | 160.45 |
| Hamm | 180761 | 226.43 |
| Herne | 157896 | 51.42 |
| Krefeld | 228550 | 137.78 |
| Köln | 1087353 | 405.02 |
| Leverkusen | 166414 | 78.87 |
| Moers | 105606 | 67.64 |
| Mönchengladbach | 268943 | 170.47 |
| Mülheim an der Ruhr | 173255 | 91.28 |
| Münster | 322904 | 303.28 |
| Neuss | 155163 | 99.52 |
| Oberhausen | 211099 | 77.09 |
| Paderborn | 155749 | 179.59 |
| Recklinghausen | 111693 | 66.5 |
| Remscheid | 112970 | 74.52 |
| Siegen | 102114 | 114.69 |
| Solingen | 161545 | 89.54 |
| Wuppertal | 358938 | 168.39 |

# Daten abfragen mit SQL

**Aufgabe 5**

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die zwischen 200000 und 250000 Einwohner haben. Name, Einwohner und Bundesland sollen aufgelistet werden, aufsteigend nach der Einwohnerzahl sortiert.

Schreibe eine entsprechende SQL-Abfrage. Nutze die land_id und den Befehl `INNER JOIN`.

## Beispiellösung

```
SELECT Grossstaedte.name, Grossstaedte.einwohner, Bundeslaender.land
FROM Grossstaedte
INNER JOIN Bundeslaender ON Grossstaedte.land_id = Bundeslaender.land_id
WHERE Grossstaedte.einwohner BETWEEN 200000 AND 250000
ORDER BY Grossstaedte.einwohner ASC;
```

**Output**

| name | einwohner | land |
|---|---|---|
| Kassel | 204687 | Hessen |
| Rostock | 210795 | Mecklenburg-Vorpommern |
| Oberhausen | 211099 | Nordrhein-Westfalen |
| Erfurt | 215675 | Thüringen |
| Lübeck | 219044 | Schleswig-Holstein |
| Mainz | 222889 | Rheinland-Pfalz |
| Krefeld | 228550 | Nordrhein-Westfalen |
| Freiburg im Breisgau | 237244 | Baden-Württemberg |
| Magdeburg | 240114 | Sachsen-Anhalt |
| Halle (Saale) | 242172 | Sachsen-Anhalt |
| Kiel | 248873 | Schleswig-Holstein |

# Daten abfragen mit SQL

**Aufgabe 6**

Aus der Tabelle Grossstaedte sollen alle Städte herausgesucht werden, die Hauptstadt eines Bundeslandes sind. Hauptstadt, Einwohner und Bundesland sollen aufgelistet werden, aufsteigend nach der Hauptstadt sortiert.

Schreibe eine entsprechende SQL-Abfrage. Nutze den Befehl `INNER JOIN`.

## Beispiellösung

```
SELECT Bundeslaender.hauptstadt, Grossstaedte.einwohner, Bundeslaender.land
FROM Bundeslaender
INNER JOIN Grossstaedte ON Grossstaedte.name = Bundeslaender.hauptstadt
ORDER BY Bundeslaender.hauptstadt ASC;
```

**Output**

| hauptstadt | einwohner | land |
|---|---|---|
| Berlin | 3782202 | Berlin |
| Bremen | 577026 | Bremen |
| Dresden | 566222 | Sachsen |
| Düsseldorf | 631217 | Nordrhein-Westfalen |
| Erfurt | 215675 | Thüringen |
| Hamburg | 1910160 | Hamburg |
| Hannover | 548186 | Niedersachsen |
| Kiel | 248873 | Schleswig-Holstein |
| Magdeburg | 240114 | Sachsen-Anhalt |
| Mainz | 222889 | Rheinland-Pfalz |
| München | 1510378 | Bayern |
| Potsdam | 187119 | Brandenburg |
| Saarbrücken | 183509 | Saarland |
| Stuttgart | 633484 | Baden-Württemberg |
| Wiesbaden | 285522 | Hessen |

# Daten abfragen mit SQL

**Aufgabe 7**

In der Tabelle Grossstaedte sollen die Großstädte gezählt werden, die zu einem Bundesland gehören. Bundesland und Anzahl der darin befindlichen Großstädte sollen aufgelistet werden, absteigend nach der Anzahl der Großstädte und aufsteigend nach dem Namen des Bundeslandes.

Schreibe eine entsprechende SQL-Abfrage. Nutze die land_id und den Befehl `INNER JOIN`.

## Beispiellösung

```
SELECT Bundeslaender.land, COUNT (Grossstaedte.land_id) AS anzahl_grossstaedte
FROM Grossstaedte
INNER JOIN Bundeslaender ON Grossstaedte.land_id = Bundeslaender.land_id
GROUP BY Bundeslaender.land
ORDER BY anzahl_grossstaedte DESC, Bundeslaender.land ASC;
```

**Output**

| land | anzahl_grossstaedte |
|---|---|
| Nordrhein-Westfalen | 30 |
| Baden-Württemberg | 9 |
| Bayern | 8 |
| Niedersachsen | 8 |
| Hessen | 6 |
| Rheinland-Pfalz | 5 |
| Sachsen | 3 |
| Brandenburg | 2 |
| Bremen | 2 |
| Sachsen-Anhalt | 2 |
| Schleswig-Holstein | 2 |
| Thüringen | 2 |
| Berlin | 1 |
| Hamburg | 1 |
| Mecklenburg-Vorpommern | 1 |
| Saarland | 1 |

# Ende-zu-Ende-Verschlüsselung

Mehr als zwei Stunden täglich verbringen Jugendliche und junge Erwachsene mit der Nutzung von Chat- oder Messengerdiensten. Das ergaben Studien zur Mediennutzung von Jugendlichen in Deutschland, die in den „Grunddaten Jugend und Medien 2020“ zusammengefasst sind. [1)]

Selbstverständlich möchte niemand, dass die mit Freundinnen und Freunden ausgetauschten Nachrichten von Dritten mitgelesen werden. Deshalb haben die Anbieter von E-Mail-, Chat- und Messengerdiensten Verfahren zur Verschlüsselung aller übertragenen Daten etabliert.

### Problem der Schlüsselverteilung

Bis in die 1970er Jahre hinein kannte man nur symmetrische Verschlüsselungsverfahren, bei denen Sender und Empfänger einer Nachricht über denselben Schlüssel verfügen müssen.

Das führt im Smartphone-Zeitalter zu der Schwierigkeit, dass für viele geheim kommunizierende Personen auch viele unterschiedliche Schlüssel benötigt werden. Die Anzahl lässt sich aus der Anzahl der Messengerdienst-Nutzer errechnen:

$$\text{Anzahl Schlüssel} = \frac{\text{Nutzer} \times (\text{Nutzer} - 1)}{2}$$

Wollten alle zwei Milliarden WhatsApp-Nutzer miteinander verschlüsselte Nachrichten austauschen, wären dafür zwei Trillionen Schlüssel nötig.

### Asymmetrische Verschlüsselung

Das so genannte Schlüsselverteilungsproblem wurde erst 1977 gelöst, als Ronald L. Rivest, Adi Shamir und Leonard M. Adleman am Massachusetts Institute of Technology das nach ihnen benannte RSA-Verfahren entwickelten.

Bei diesem ersten asymmetrischen Verschlüsselungsverfahren verfügt jeder Nutzer über ein eigenes Schlüsselpaar: einen öffentlichen und einen privaten Schlüssel.

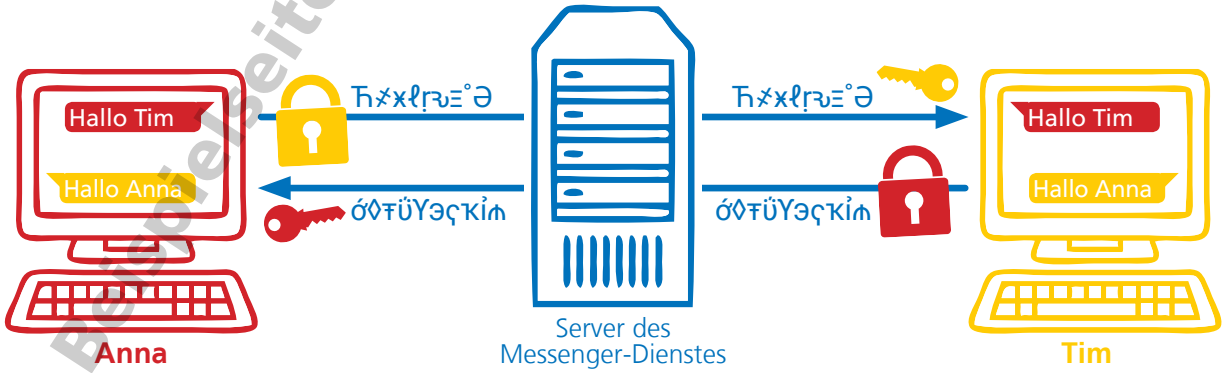
Asymmetrische Verschlüsselungsverfahren benötigen mehr Rechenleistung für das Ver- und Entschlüsseln der Daten, aber sie sind sicherer als symmetrische Verschlüsselungsverfahren. Sie ermöglichen eine sichere Kommunikation im Internet, ohne immense Kosten und Komplexität zu verursachen.

Mit den öffentlichen Schlüsseln können Daten nur verschlüsselt werden. Die öffentlichen Schlüssel aller Nutzer und Nutzerinnen eines Messengerdienstes sind in einem Verzeichnis abgelegt, auf das alle Nutzer und Nutzerinnen des Messengerdienstes zugreifen können.

Möchte Anna eine Nachricht an Tim senden, verschlüsselt sie die Nachricht mit dem öffentlichen Schlüssel von Tim. Die Nachricht ist auf dem Weg zu Tim von niemandem zu entziffern. Nur der zu Tims öffentlichem Schlüssel passende private Schlüssel kann die Nachricht entschlüsseln. Die Antwort an Anna verschlüsselt Tim mit Annas öffentlichem Schlüssel. Und nur sie kann mit ihrem privaten Schlüssel Tims Nachricht entschlüsseln.

Da das Ver- und Entschlüsseln nur beim Sender und Empfänger einer Nachricht möglich ist, spricht man von einer Ende-zu-Ende-Verschlüsselung. Beim Austauschen von Nachrichten bemerkt man davon nichts, denn Messengerdienste wie WhatsApp führen das Ver- und Entschlüsseln vollständig ohne unser Zutun durch.

**Ende-zu-Ende-Verschlüsselung bei einem Messengerdienst**




[1)] https://www.br-online.de/jugend/izi/deutsch/Grunddaten_Jugend_Medien.pdf (Stand Juni 2021)

# Ende-zu-Ende-Verschlüsselung

**Aufgabe 1**
Stell dir vor, ihr würdet in eurer Klasse über einen Messengerdienst symmetrisch verschlüsselte Nachrichten austauschen.

Wie viele unterschiedliche symmetrische Schlüssel wären notwendig, damit jede Schülerin und jeder Schüler mit allen anderen Schülerinnen und Schülern deiner Klasse Nachrichten austauschen kann?

$$\text{Anzahl Schlüssel} = \frac{\text{Nutzer} \times (\text{Nutzer} - 1)}{2}$$

Für eine Klasse mit 24 Schülerinnen und Schülern bedeutet das

$$\text{Anzahl Schlüssel} = \frac{24 \times (24 - 1)}{2} = 276$$

**Aufgabe 2**
Beschreibe den Ablauf der Ende-zu-Ende-Verschlüsselung bei einem Messengerdienst.

Eine Nachricht von A an B wird mit dem öffentlichen Schlüssel von B verschlüsselt.

B kann die Nachricht mit dem eigenen privaten Schlüssel entschlüsseln und lesen.

Die Antwort an A verschlüsselt B entsprechend mit dem öffentlichen Schlüssel von A. A kann die Nachricht mit dem eigenen privaten Schlüssel entschlüsseln und lesen.

**Aufgabe 3**
Warum kann der Betreiber des Messengerdienstes die auf seinem Server zwischengespeicherten Nachrichten nicht lesen?

Mit dem öffentlichen Schlüssel können Nachrichten nur verschlüsselt werden.

Zum Entschlüsseln benötigt man den privaten Schlüssel des Empfängers. Diesen Schlüssel kennt der Betreiber des Messengerdienstes nicht. Daher kann er die auf seinem Server zwischengespeicherten Nachrichten nicht lesen.

**Aufgabe 4**
Worin besteht der wichtigste Unterschied zwischen symmetrischen und asymmetrischen Verschlüsselungen?

Bei einer symmetrischen Verschlüsselung werden alle übertragenen Daten in beiden Richtungen mit ein und demselben Schlüssel ver- und entschlüsselt.

Bei einer asymmetrischen Verschlüsselung werden unterschiedliche Schlüssel für das Ver- und Entschlüsseln der Daten genutzt. Meist handelt es sich dabei um einen öffentlichen und einen privaten Schlüssel.