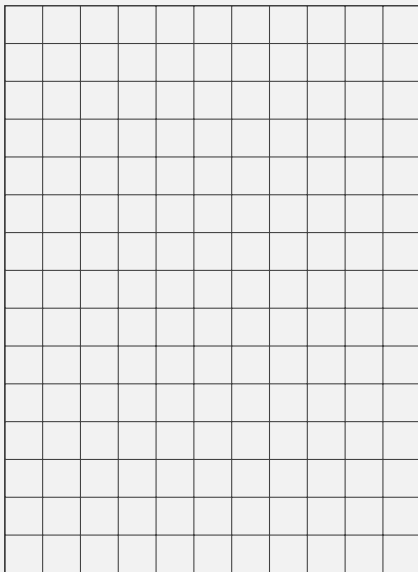


## 1 Wdh: Klassen und Objekte

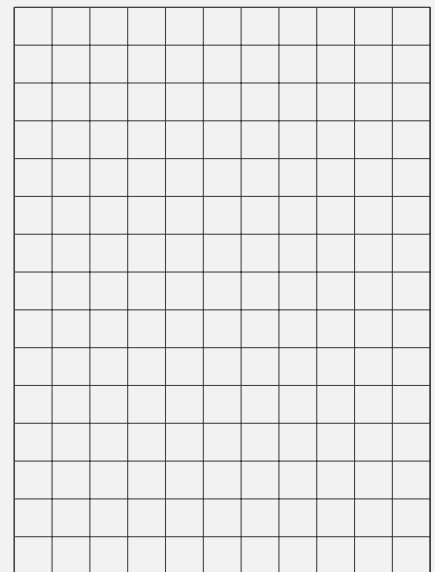


\_\_\_\_\_ repräsentieren **Gegenstände** in einem Computerprogramm.  
\_\_\_\_\_ sind der **Bauplan**, der festlegt, welche **Eigenschaften** (  
\_\_\_\_\_ ) und **Fähigkeiten** ( \_\_\_\_\_ ) einer bestimmten  
Objektart gespeichert werden sollen. Man stellt sie dar mit:

### Klassenkarte



### Objektkarte

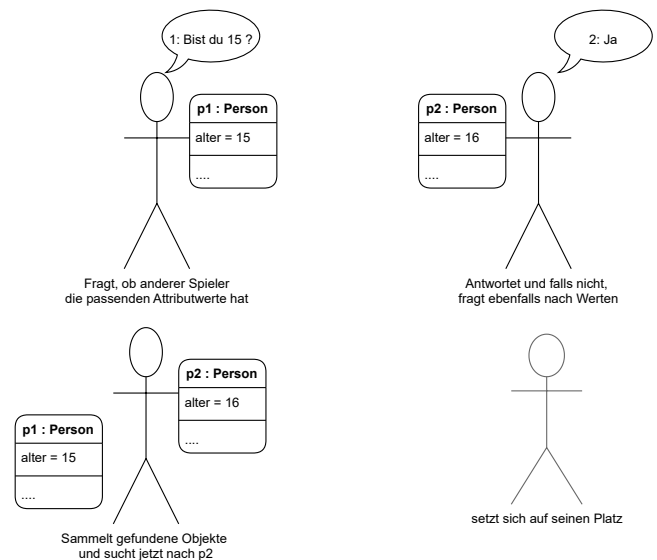


❗ Generell kann man Objektkarten mit oder ohne Methoden zeichnen, solange man es insgesamt einheitlich macht. Wir zeichnen sie daher immer ohne Methoden.



## Objektkarten Memory

- Erstelle auf einem Blatt eine Objektkarte der Klasse Person zu dir selbst. → **3x falten**
- Gib deine Objektkarte bei der Lehrkraft ab. → Objektkarten werden gemischt.
- Ziehe eine Objektkarte und versuche, das zugehörige Objekt zu finden.
  - Frage deine:n Gegenüber dafür, ob die Attributwerte auf deiner gezogenen Karte auf sie/ihn zutreffen.
  - Ihr dürft euch nicht gegenseitig die Objektkarten zeigen!
  - Wer gefunden wurde, gibt seine aktuelle Objektkarte weiter und setzt sich.
  - Der/Die Finder:in sammelt alle gefundenen Objekte.






### 3 SQL Spickzettel



Folgender SQL-Spickzettel enthält alle SQL-Grundlagen der 9. Klasse. Ihr dürft (sollt!) ihn bei allen SQL-Aufgaben benutzen. Über das Vorlagensymbol oben könnt ihr den Spickzettel als eigenes PDF öffnen.

Übrigens: **SQL** ist die Abkürzung für **S**tructured **Q**uery **L**anguage, was auf Deutsch etwa Strukturierte Abfrage Sprache heißt.

Inf 9 Grundlagen	Spickzettel SQL	
---------------------	-----------------	---

**SELECT**

**FROM**

**WHERE**

**GROUP BY**

**HAVING**

**ORDER BY**

Spaltenliste

Tabelle

Bedingung

Spaltenliste

Bedingung

Spaltenliste

- DISTINCT vermeidet Duplikate.
- Aggregatfunktionen (COUNT, SUM, MAX, MIN, AVG) für Berechnungen
- SELECT \* für "alle Spalten"
- AS Aliasname

- Wird meist mit **Vergleichen** (<, <=, =, >, >=) formuliert.
- Verknüpfung von mehreren Vergleichen mit **logischen Funktionen** (AND, OR, NOT)

- ASC für aufsteigend (Standard)
- DESC für absteigend

Im Detail gilt:

Grundlegende SQL-Abfrage		
SELECT	Es muss mindestens ein Spaltenname angegeben werden. Die entsprechende(n) Spalte(n) sind dann Teil der Ergebnistabelle. SELECT * bewirkt, dass alle Spalten angezeigt werden.	
DISTINCT	Duplikate von Datensätzen werden nicht angezeigt.	
AS	Eine Spalte in der Ergebnistabelle kann anders benannt werden als in der Ausgangstabelle. Dies ist vor allem bei der Verwendung von Aggregatfunktionen hilfreich.	
FROM	Hier muss angegeben werden, aus welcher Tabelle die Informationen für die Abfrage genommen werden sollen.	
ORDER BY	Die Ergebnistabelle wird nach der oder den angegebenen Spalten sortiert. Standardmäßig wird aufsteigend sortiert. Mit dem Zusatz DESC bzw. ASC wird absteigend bzw. aufsteigend sortiert.	
Beispiele	<pre>SELECT DISTINCT kontinent AS "enthaltene Kontinente" FROM Land</pre> <pre>SELECT name, flaeche, hauptstadt FROM LAND ORDER BY flaeche DESC</pre>	

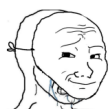
Auswahl von Datensätzen über Bedingungen	
WHERE	In der Ergebnistabelle werden nur die Datensätze (Zeilen) angezeigt, welche die angegebene Bedingung erfüllen. Eine Bedingung wird mit einem Vergleich formuliert. Neben den typischen Vergleichsoperatoren wie <, <=, =, >, >=, usw. sind insbesondere auch IS NULL und LIKE wichtig. Mehrere Vergleiche können durch die logischen

	Funktionen AND, OR und NOT verknüpft werden. Ggf. müssen die einzelnen Ausdrücke dabei sinnvoll geklammert werden
Beispiel	<pre>WHERE jahr &gt; 2015 AND laufzeit &lt;= 90 AND NOT fsk = 18</pre>
LIKE	<p>Kann in einer Bedingung zur Mustererkennung von Einträgen verwendet werden. Folgende zwei Platzhalter (wildcards) werden häufig eingesetzt:</p> <ul style="list-style-type: none"> <li>% steht für beliebig viele Zeichen, auch keines (* bei MS Access)</li> <li>_ für genau ein beliebiges Zeichen (? bei MS Access)</li> </ul> <p>Beispiele:</p> <ul style="list-style-type: none"> <li>WHERE titel LIKE "You%" – findet alle Titel die mit "You" beginnen</li> <li>Groß-/Kleinschreibung wird nicht berücksichtigt</li> <li>WHERE titel LIKE "%love%" – findet alle Titel die "love" enthalten</li> <li>WHERE titel LIKE "L____" – findet alle Titel die mit L beginnen und genau 4 Zeichen lang sind</li> </ul>
NULL	Bedeutet, dass kein Wert in einer Zelle eingetragen ist.
IS NULL	Überprüft (in einer Bedingung), ob kein Wert in einer Zelle eingetragen ist.

Aggregatfunktionen	
AVG	Berechnet den Durchschnitt aller Werte einer Spalte.
COUNT	Gibt die Anzahl der Einträge einer Spalte aus.
MAX bzw. MIN	Gibt das Maximum bzw. Minimum aller Werte einer Spalte aus.
SUM	Berechnet die Summe aller Werte einer Spalte.
Beispiel	<pre>SELECT COUNT(*) AS "Anzahl afrikanischer Länder" FROM Land WHERE kontinent = "Afrika"</pre>

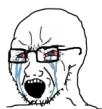
Gruppierung	
GROUP BY	Datensätze mit demselben Wert in der angegebenen Spalte werden gruppiert. Gruppierungen sind nur in Kombination mit Aggregatfunktionen sinnvoll.
HAVING	An gruppierte Datensätze werden Bedingungen mit HAVING formuliert.
Beispiel	<pre>SELECT fsk, MIN(laufzeit) FROM Film WHERE genre1="Filmkomödie" OR genre2="Filmkomödie" GROUP BY fsk HAVING fsk &lt;16</pre>

SQL keywords should be  
in **lower case!**



```
select name, id
from products
where discount = 0
order by price asc;
```

Noooo, they must be  
in **upper case!**



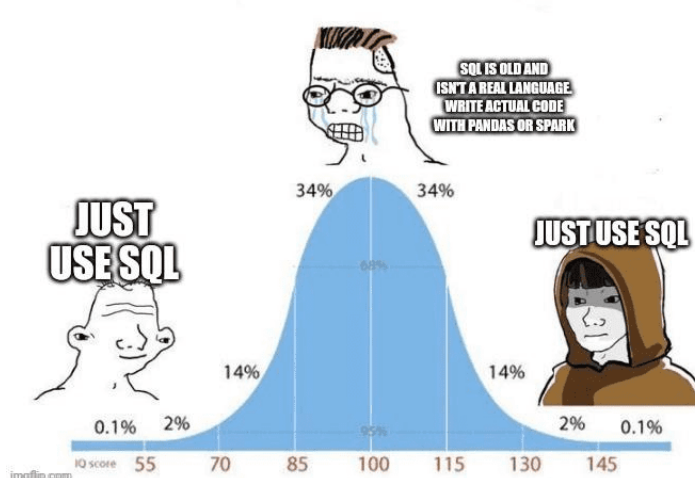
```
SELECT name, id
FROM products
WHERE discount = 0
ORDER BY price ASC;
```



```
sELeCt nAmE, iD
fRoM PrOdUcTs
WhErE dIsCoUnT = 0
OrDeR bY pRiCe AsC;
```

'Sarcastic Query Language' • by u/casperdewith

SQL Schlüsselwörter wie SELECT, WHERE etc. sind nicht case-sensitive. Groß-/Kleinschreibung ist also egal.



sql-island.informatik.uni-kl.de/

- ❗ Den Datentyp Character gibt es in den meisten Datenbanksystemen nicht. Wir verwenden daher immer String (=Text).

- teilt die Tabellen der Datenbank mit Klassenkarten dar:

- 4



In dieser Aufgabe geht es immer um die Tabelle **land**, deren erste Datensätze du hier siehst:

id	name	einwohner	flaeche	hauptstadt
1	Deutschland	83.24	358	Berlin
2	Frankreich	67.39	544	Paris
3	Brasilien	212.60	8516	Rio de Janeiro
...	...	...	...	...

**Welche SQL-Abfrage (rechte Seite) führt zu welcher Ergebnistabelle (linke Seite)? Ordne richtig zu!**

1) Zeige alle Spalten der Tabelle land.

i) SELECT name  
FROM land  
ORDER BY name DESC;

2) Zeige die Spalten name und hauptstadt der Tabelle land.

ii) SELECT name  
FROM land  
WHERE name LIKE 'D%'  
OR name LIKE '%d';

3) Zeige die durchschnittliche Einwohnerzahl aller Länder.

iii) SELECT COUNT(\*)  
FROM land  
WHERE name LIKE '%land';

4) Zeige die Namen aller Länder in alphabetisch absteigender Reihenfolge.

iv) SELECT \*  
FROM land;

5) Zeige die Hauptstädte der Länder, deren Einwohnerzahl größer als 50 Mio ist.

v) SELECT name  
FROM land  
WHERE flaeche >= 100  
AND flaeche <= 999;

6) Zeige die Anzahl aller Länder, deren Name mit 'land' endet.

vi) SELECT name, einwohner  
FROM land  
ORDER BY einwohner DESC  
LIMIT 3;

7) Zeige die Namen aller Länder, deren Fläche zwischen 100 und 999 Tausend km<sup>2</sup> liegt.

vii) SELECT AVG(einwohner)  
FROM land;

8) Zeige die Namen der Länder, die mit 'D' beginnen oder mit 'd' aufhören.

viii) SELECT name, hauptstadt  
FROM land;

9) Zeige die Namen der drei Länder mit der größten Einwohnerzahl.

ix) SELECT hauptstadt  
FROM land  
WHERE einwohner > 50;







## Tabellenbeziehungen

1. Visualisiere (mit Bleistift), wer Häuptling in welchem Dorf ist.
2. Überlege, wie du allgemein für diese zwei Tabellen darstellen kannst, wie sie (und ihre Spalten) miteinander in Beziehung stehen.

**SELECT \* FROM dorf**

dorfnr	name	haeuptling
1	Affenstadt	1
2	Gurkendorf	6
3	Zwiebelhausen	7

**SELECT \* FROM Bewohner**

bewohnernr	name	dorfnr	geschlecht	beruf	gold	status
1	Paul Backmann	1	m	Baecker	850	friedlich
2	Ernst Peng	3	m	Waffenschmied	280	friedlich
3	Rita Ochse	1	w	Baecker	350	friedlich
4	Carl Ochse	1	m	Kaufmann	250	friedlich
5	Dirty Dieter	3	m	Schmied	650	boese
6	Gerd Schlachter	2	m	Metzger	4850	boese
7	Peter Schlachter	3	m	Metzger	3250	boese
8	Arthur Schneiderpaule	2	m	Pilot	490	gefangen

### Tabellenbeziehung im Klassendiagramm

1. Ergänze das Klassendiagramm entsprechend der beiden Tabellen oben.
2. Wie kann man die Beziehungen zwischen den beiden Tabellen im Klassendiagramm darstellen?  
Tipp: Unsere Überlegungen von oben, helfen dabei.

Dorf

Bewohner



## 4 Tabellenbeziehungen: Fremdschlüssel



Wenn Datensätze mittels Primärschlüssel in einer anderen Tabelle verwendet werden, spricht man dort von einem Fremdschlüssel. Im **Tabellenschema** werden die \_\_\_\_\_ durch - (manchmal auch .) markiert. Ein Beispiel in SQL-Island ist der Häuptling eines Dorfes, der in der Tabelle Dorf mittels bewohnernr eingetragen wird. Die **bewohnernr** ist hierbei \_\_\_\_\_ in der **Tabelle Bewohner** und \_\_\_\_\_ in der **Tabelle Dorf** (heißt hier aber **haeuptling**).

## 5 Tabellenbeziehungen im Klassendiagramm



TabelleA
int id
String spalte1
...

TabelleB
int id
String spalte1
...

## 6 Kardinalitäten



Die Kardinalität beschreibt, wie viele Objekte auf jeder Seite einer Beziehung stehen können. Es gibt folgende Arten:

- **1:1**, z.B. \_\_\_\_\_ Häuptling pro Dorf, der auch nur in einem Dorf Häuptling ist.
- **1:n**, z.B. jeder Bewohner wohnt in einem Dorf, das aber \_\_\_\_\_ Bewohner hat.
- **m:n**, z.B. \_\_\_\_\_ Lehrer pro Schulklasse + \_\_\_\_\_ Schulklassen pro Lehrer (in Datenbanken nicht direkt umsetzbar, dazu später mehr).

[illegible]

## 7 Kreuzprodukt / Join



Möchte man Daten aus zwei Tabellen mit Beziehung zueinander abfragen, gibt man beide Tabellen **mit Komma getrennt nach FROM** an.

Die SQL-Abfrage bildet dann das \_\_\_\_\_ der Tabellen. Die Ergebnistabelle enthält \_\_\_\_\_ von Datensätzen beider Tabellen (**Merkregel:** \_\_\_\_\_).

Um nur zusammengehörige Datensätze (also solche, die miteinander in Beziehung stehen, z.B. eine Bewohner mit seinem Dorf) auszuwählen, ergänzt man als **Selektion** eine **Gleichheitsbedingung** zwischen Fremd- und zugehörigem

\_\_\_\_\_. Dann spricht man von einem \_\_\_\_\_.

Zum Beispiel kann man in SQL-Island die Daten aller Dörfer und ihrer zugehörigen Häuptlinge so ausgeben:

```
SELECT *
FROM Dorf, Bewohner
WHERE Dorf.haeuptling = Bewohner.bewohnernr
```

## 8 Join Beispiel



Lehrkraft		
id	kuerzel	schule
1	Her	MTG
2	Ext	Dante

```
SELECT *
FROM Lehrkraft, Schule
WHERE Lehrkraft.schule = Schule.id
```

Schule	
id	ort
MTG	Haidh.
Dante	Sendl.

**Ergebnistabelle des Kreuzprodukts:**

id	kuerzel	schule	id	ort
1	Her	MTG	MTG	Haidh.
2	Ext	Dante	MTG	Haidh.
1	Her	MTG	Dante	Sendl.
2	Ext	Dante	Dante	Sendl.

**Ergebnistabelle des Joins**

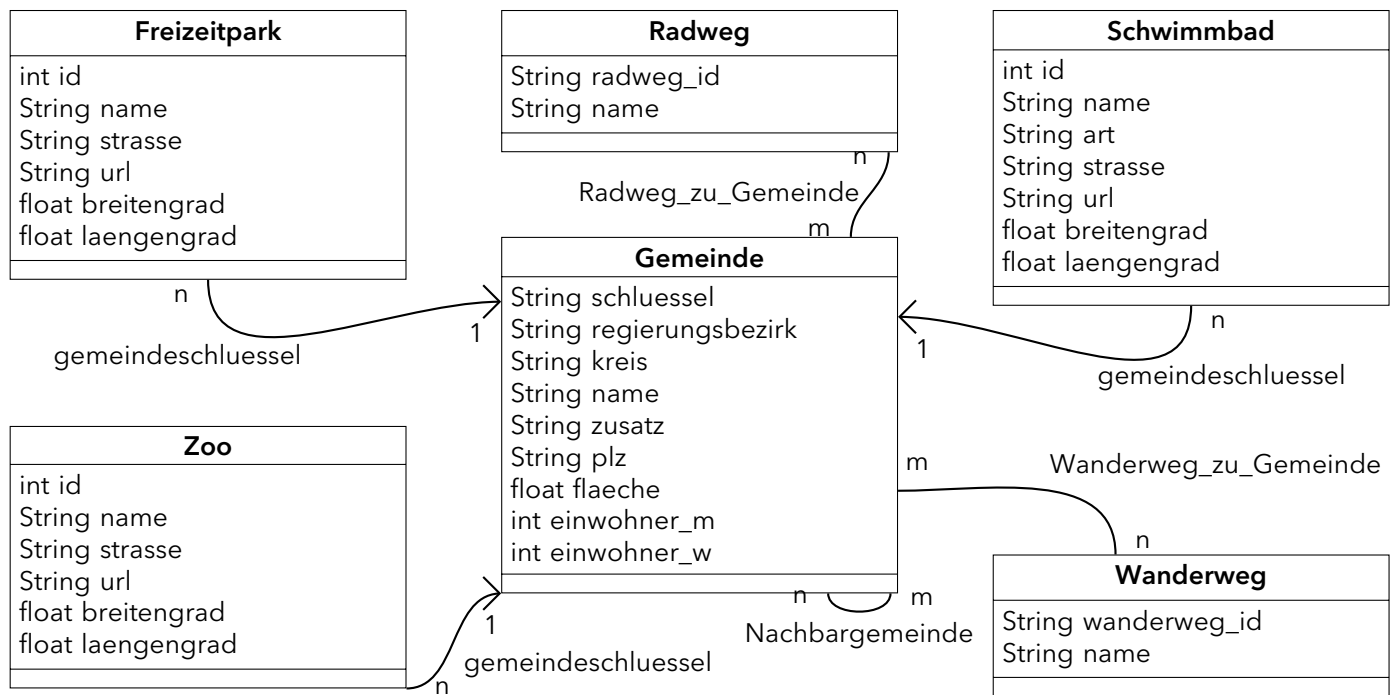
id	kuerzel	schule	id	ort
1	Her	MTG	MTG	Haidh.
2	Ext	Dante	Dante	Sendl.

## SQL mit Kreuzprodukt und Join

Bearbeite diese Aufgabe auf [artemis.tum.de](https://artemis.tum.de). Du bekommst eine automatische Rückmeldung, ob deine Abgabe korrekt ist. Alle Aufgaben beziehen sich auf die Datenbank mit unten stehendem Klassendiagramm. Eine Online-Version gibt es unter [www.dbiu.de/bayern/](https://www.dbiu.de/bayern/), dort ist auch das Tabellenschema zu finden.

Gib immer genau die geforderten Daten aus und nicht mehr. Sortiere nicht, wenn du nicht dazu aufgefordert wirst.

**Notiere unten anschließend deine korrekten SQL-Abfragen unten.**



Verändere die SQL-Abfrage so, dass die Namen und Internetadressen (=url) aller Zoos und der Name und Regierungsbezirk der jeweiligen Gemeinde ausgegeben wird:

**SELECT Zoo.name, Gemeinde.name** \_\_\_\_\_

**FROM Zoo, Gemeinde**

Verändere die SQL-Abfrage so, dass die Namen und Straßen aller Freizeitparks und die Namen der jeweils zugehörigen Gemeinde ausgegeben wird.

**SELECT Freizeitpark.name, Gemeinde.name** \_\_\_\_\_

**FROM Freizeitpark, Gemeinde**



