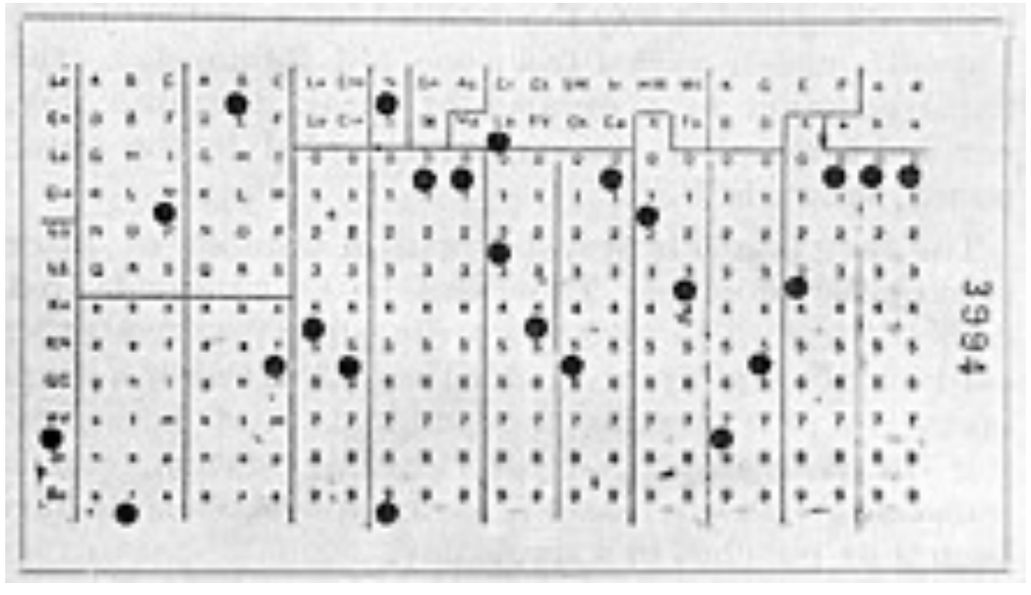
INFORMATIK KLASSE 9

LERNBEREICHE:
DATEN MODELLIEREN – DATENBANKSYSTEME
DATEN DARSTELLEN – INFORMATIKPROJEKTE

1 Geschichte der Datenbanken

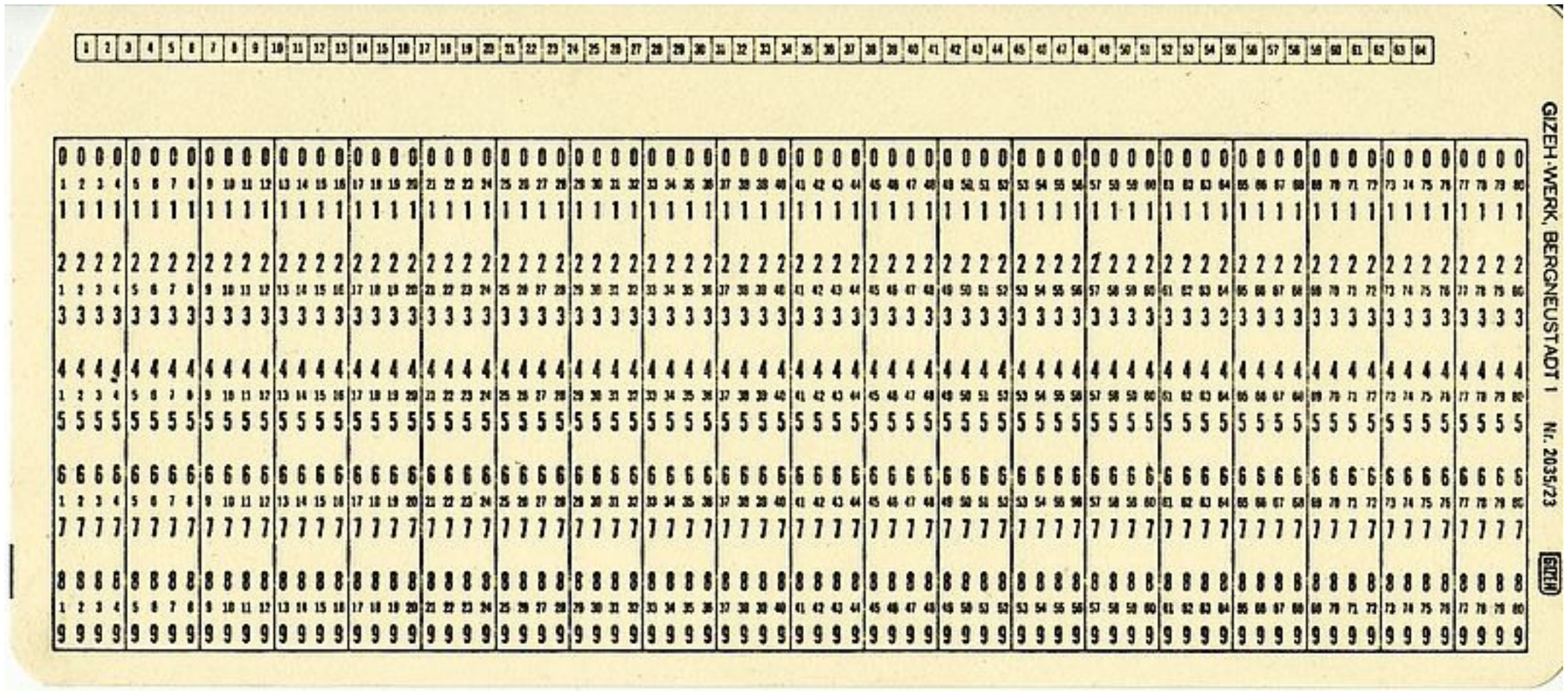
1.1 Lochkarten

Lochkarten als Speicher für Webstühle



Hollerith-Lochkarte

Lochkarte – IBM Standard



Ursprünglich war nur ein Loch in jeder der 80 Spalten erlaubt. Es könnten die Ziffern von 0-9 und die Vorzeichen + und - gestanzt und damit gespeichert werden.

Ab 1964 könnten die 80 Spalten in den 10 Zeilen bis zu 6 Lochungen enthalten - damit konnten pro Spalte 265 verschiedene Zeichen kodiert werden. Pro Karte konnte also ein Text bis zu 80 Zeichen gespeichert werden. (Wikipedia: 12/2019, Lochkarte)

1.2 Datensammlungen in Dateien (CSV–Format)

- Ein ursprüngliches Format, hat sich bis heute erhalten - das CSV-Format (engl. Comma-separated values)
- Einsatz heute:
  - Austausch von Daten für Datenbanken oder Tabellenkalkulationsprogramme
  - in UNIX-Betriebssystemen als Benutzerdatenbank

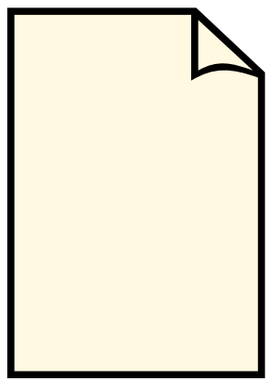
1.2 Datensammlungen in Dateien (CSV-Format)

Aufbau einer CSV-Datei am Beispiel (Städteinformationen)

```
#Stadt;Bundesland;Landkreis;Fläche[km^2];Einwohnerzahl
Großröhrsdorf;Sachsen;Bautzen;40,92;9510
Pulsnitz;Sachsen;Bautzen;26,75;7467
Ohorn;Sachsen;Bautzen;11,99;2453
```

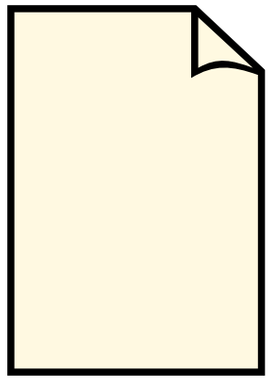
Dieser Text kann mit jedem beliebigen Texteditor erstellt werden und in jede Tabellenkalkulation oder Datenbank eingelesen werden.-> (Austauschformat)

CSV <-> Tabellenkalkulation



staedte.csv

```
#Stadt;Bundesland;Landkreis;Fläche[km^2];Einwohnerzahl
Großröhrsdorf;Sachsen;Bautzen;40,92;9510
Pulsnitz;Sachsen;Bautzen;26,75;7467
Ohorn;Sachsen;Bautzen;11,99;2453
```



staedte.ods

Tabellenkalkulation (Daten sortieren)

A1 | f_x | Σ | = | #Stadt

	A	B	C	D	E
1	#Stadt	Bundesland	Landkreis	Fläche[km^2]	Einwohnerzahl
2	Großröhrsdorf	Sachsen	Bautzen	40,92	9510
3	Pulsnitz	Sachsen	Bautzen	26,75	7467
4	Ohorn	Sachsen	Bautzen	11,99	2453
5					
6					

	A	B	C	D	E
1	#Stadt	Bundesland	Landkreis	Fläche[km^2]	Einwohnerzahl
2	Großröhrsdorf	Sachsen	Bautzen	40,92	9510
3	Ohorn	Sachsen	Bautzen	11,99	2453
4	Pulsnitz	Sachsen	Bautzen	26,75	7467

1.3 Begriffe (Zusammenfassung)

CSV-Datei:

- Datei mit durch Trennzeichen getrennten Daten (Comma-separated value)
- Austauschformat

CSV-Datei / Tabelle:

- Struktur der Datenspeicherung
- jede Zeile enthält einen Datensatz (alle Attributwerte eines Objektes)

Datensatz:

- Attributwerte zu einem Objekt

2 Datenbanken

2.1 Was wissen wir schon?

Wir erstellen ein Mindmap!

Datenbank

2.2 Herkömmliche Datensammlung in Dateien

- Speicherung von Informationen in Dateien, z.B. Mitgliederlisten im Sportverein

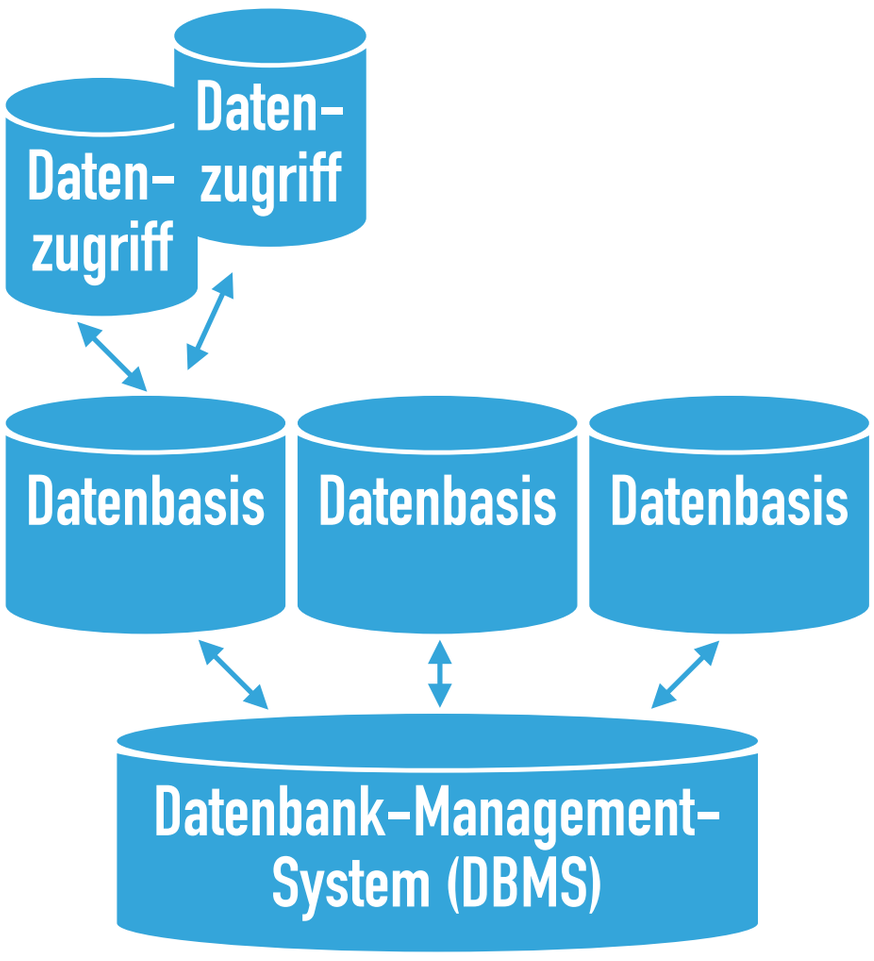
Welche Nachteile erkennst Du bei dieser Variante?

- mehrfache Speicherung (Redundanz) der Daten in verschiedenen Dateiformaten. Fehler oder Unklarheiten entstehen durch unterschiedliche Angaben (Anomalie) in den Dateien, z.B. unterschiedliche Preise
- Aktualisierung aufwendig, denn Änderungen muss in verschiedenen Formaten erfolgen.
- Nur eine Instanz kann schreibend zugreifen, sonst entsteht Datenverlust durch das Überschreiben.
- Daten können nur von einem Programm oder einem Programmtyp gelesen werden.

2.3 Vorteile von Datenbanken

- Zugriff durch eine Vielzahl an Nutzern
- Bereitstellen der Daten an verschiedene Programme (z.B. Webseiten, App, Textverarbeitung - Serienbrief, Tabellenkalkulation - Datenauswertung, Diagramme usw.)
- Sichere und zuverlässige Veränderung der Daten, z.B. Buchung von Geldbeträgen.
- Optimierter Speicherplatz
- zentrale Datenspeicherung mit Zugriff von mehreren Rechnern.

2.4 Allgemeiner Aufbau einer Datenbank



Jeder Nutzer (Programm) bekommt oder sendet seine gewünschten Daten

Jede Datenbasis enthält eine strukturierte Sammlung von Daten, meist in Tabellenform.

Software zur Verwaltung der Ein- und Ausgaben und zur Organisation der Datenspeicherung.

2.4 Allgemeiner Aufbau einer Datenbank

Nur das DBMS speichert die Daten!

Alle Nutzer oder Programme lesen oder speichern die Daten durch Zugriff auf die Datenbank.

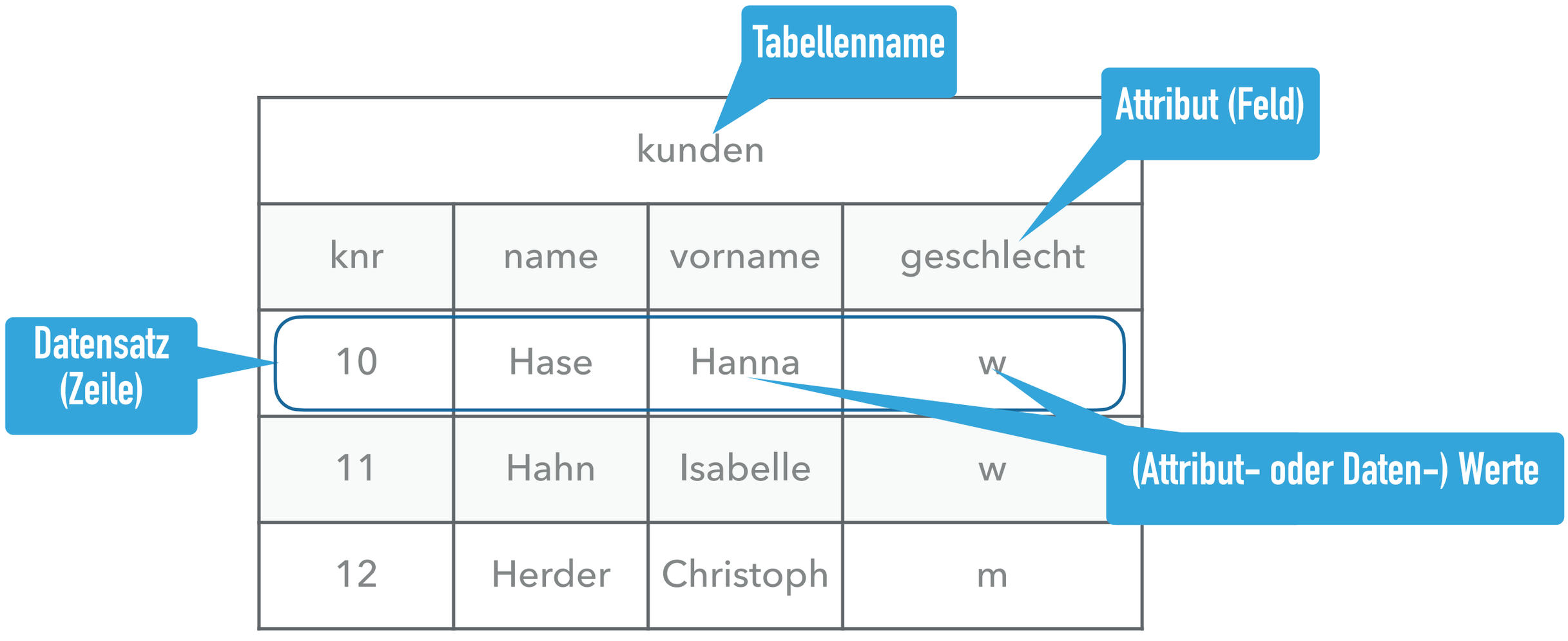
2.5 Relationale Datenbank

2.5.1 Bekannte relationale Datenbankmanagementsysteme

- Microsoft Access
- LibreOffice Base
- Oracle
- MySQL
- PostgreSQL

2.5.2 Aufbau einer Relationalen Datenbank (Bsp. Übungsdatenbank)

In relationalen Datenbanken werden die Daten in mehreren, verknüpften Tabellen gespeichert.



kunden

knr	name	vorname	geschlecht
10	Hase	Hanna	w
11	Hahn	Isabelle	w
12	Herder	Christoph	m

Entsprechungen zu Klassen und Objekten

- Die **Tabelle** entspricht einer **Klasse.**
- Die Spalten entsprechen den **Attributen** eines Objektes**.**
- Eine **Zeile (Datensatz)** entspricht dem **Objekt.** Die Zeile setzt sich aus mehreren **Attributwerten** zusammen.

kunden			
knr	name	vorname	geschlecht
10	Hase	Hanna	w
11	Hahn	Isabelle	w
12	Herder	Christoph	m

2.5.3 Primärschlüssel (Bsp. Übungsdatenbank)

Jedes Objekt muss durch ein oder mehrere **Attributwerte** eindeutig sein. Diese besonderen Attribute werden **Primärschlüssel** genannt. Oft wird ein Attribut aus fortlaufenden Zahlen als Attributwert verwendet. (z.B. Kundennummer)

In der Tabelle Fahrrad ist dies „fnr“ - die Fahrradnummer, die an jedem Fahrrad angebracht wird.

Fahrraeder

fnr	hersteller	modell	rahmen	artnr
1	Staiger	Orkan	25	02
2	Raleigh	Executive	27	02
3	Panasonic	FirstClass	28	01
...				
37	Dawes	OnOffRoad	28	03

2.5.4 Verbindung der Datensätze zwischen den Tabellen (Beziehung)

Fahrraeder

fnr	hersteller	modell	rahmen	artnr
1	Staiger	Orkan	25	02
2	Raleigh	Executive	27	02
3	Panasonic	FirstClass	28	01
...				
37	Dawes	OnOffRoad	28	03

Fremdschlüssel

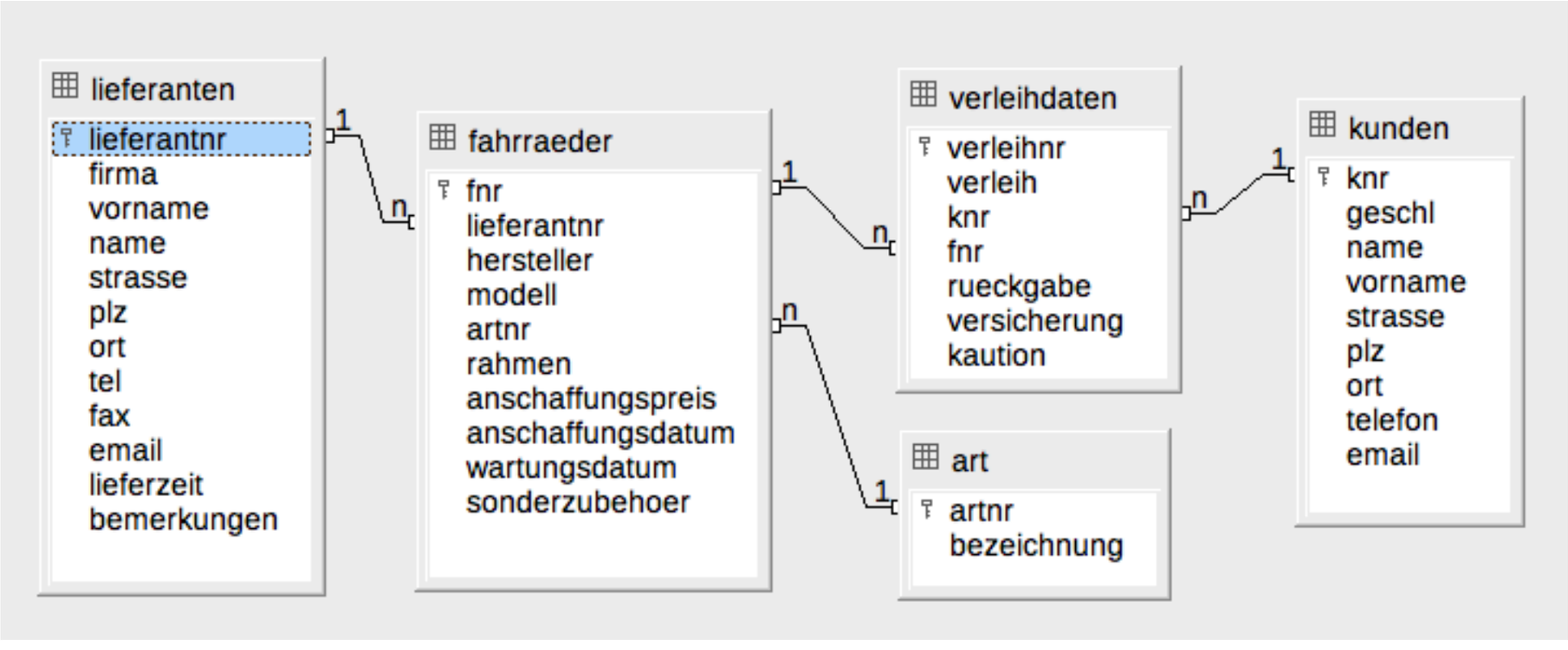
art

artnr	bezeichnung
01	Mountain-Bike
02	Tourenrad
03	Citybike
04	Rennrad
05	Tandem

Primärschlüssel

Zwei Tabellen werden über die Beziehung „Fremdschlüssel“ und „Primärschlüssel“ miteinander verbunden!

Übungsdatenbank Fahrradverleih



Darstellungen von Beziehungen zwischen Tabellen

Das Entity-Relationship-Modell

- Das Entiy-Relationship-Modell dient der Planung und der Darstellung einer Datenbank.

- Was ist eine Entität?

    - Die Entität ist das „Ding der Betrachtung“. Von diesem Ding sollen später mehrere Objekte gespeichert werden. Z.B. Schüler mit den Objekten Susi, Franz und Peter aus der Klasse 9a. Die Entität ist also die Klasse der Objekte.

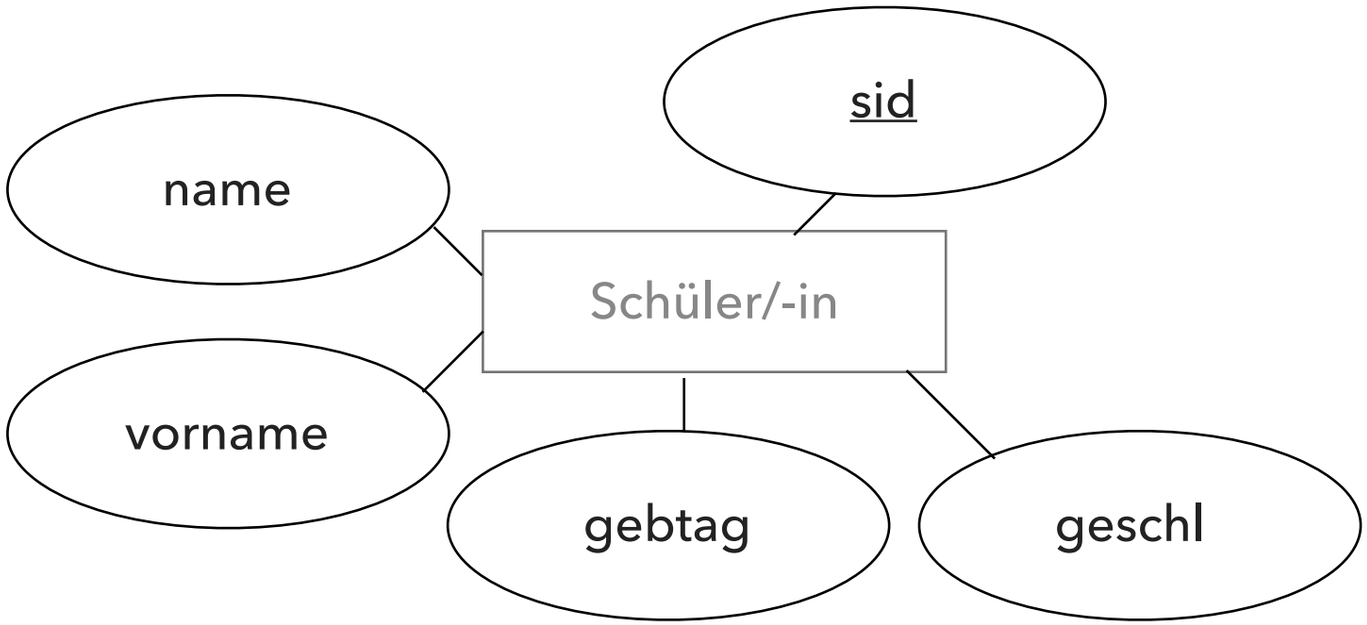
    - In einer Relationalen Datenbank wird die Klasse (Entität) in Form einer Tabelle erstellt. Die Objekte sind die einzelnen Datensätze (Zeilen) in der Tabelle.

Das Entity-Relationship-Modell

- Darstellung der Klasse (Entität):

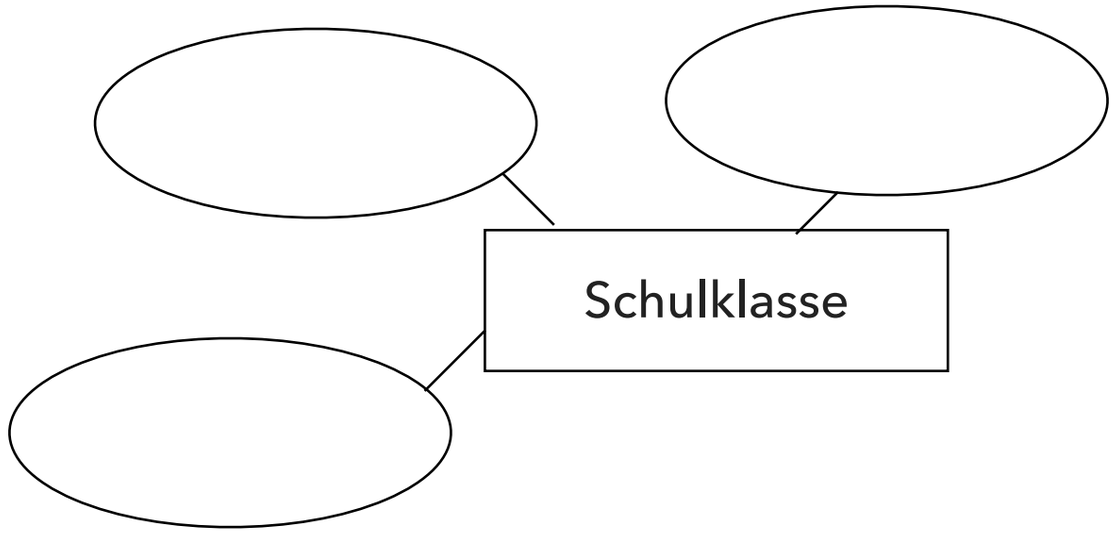
Das Entity-Relationship-Modell

- Attribute der Klasse:



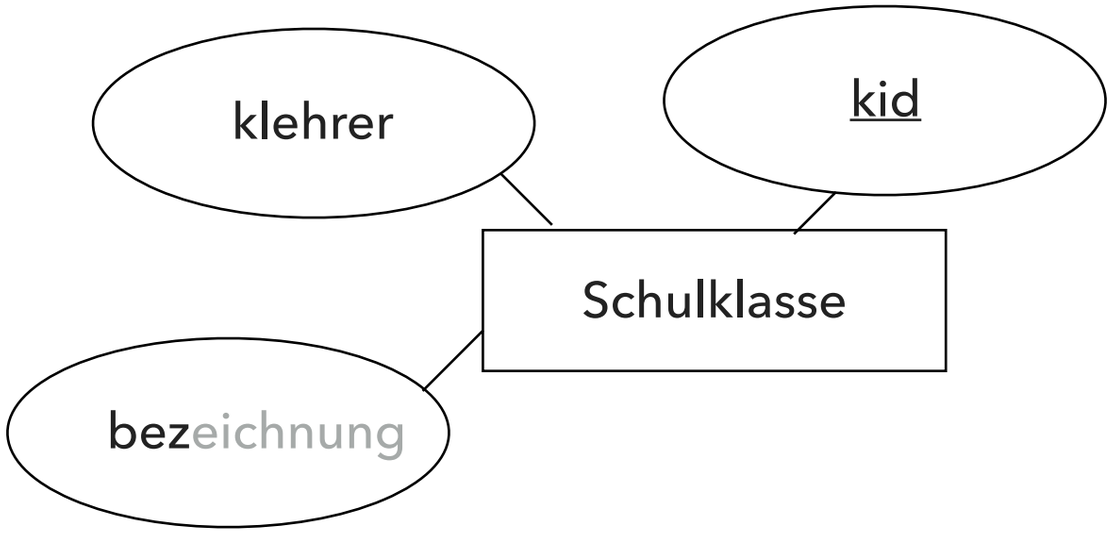
Das Entity-Relationship-Modell

- Übung



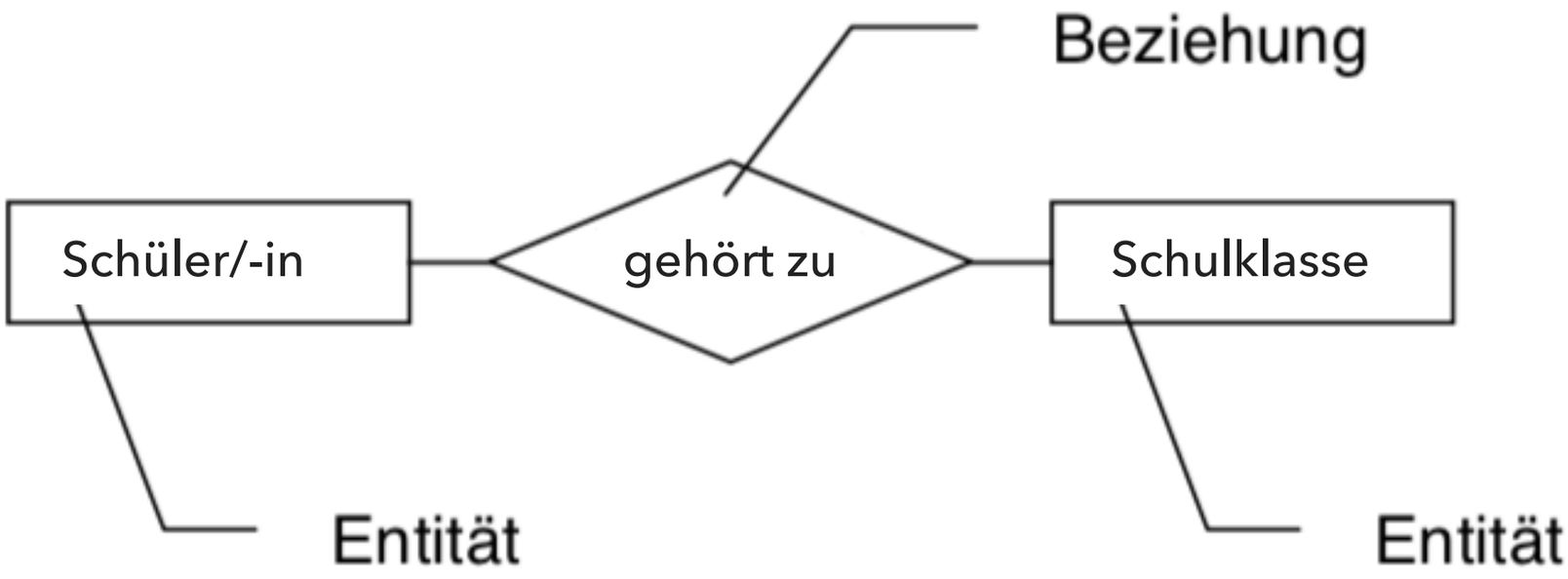
Das Entity-Relationship-Modell

- Übung



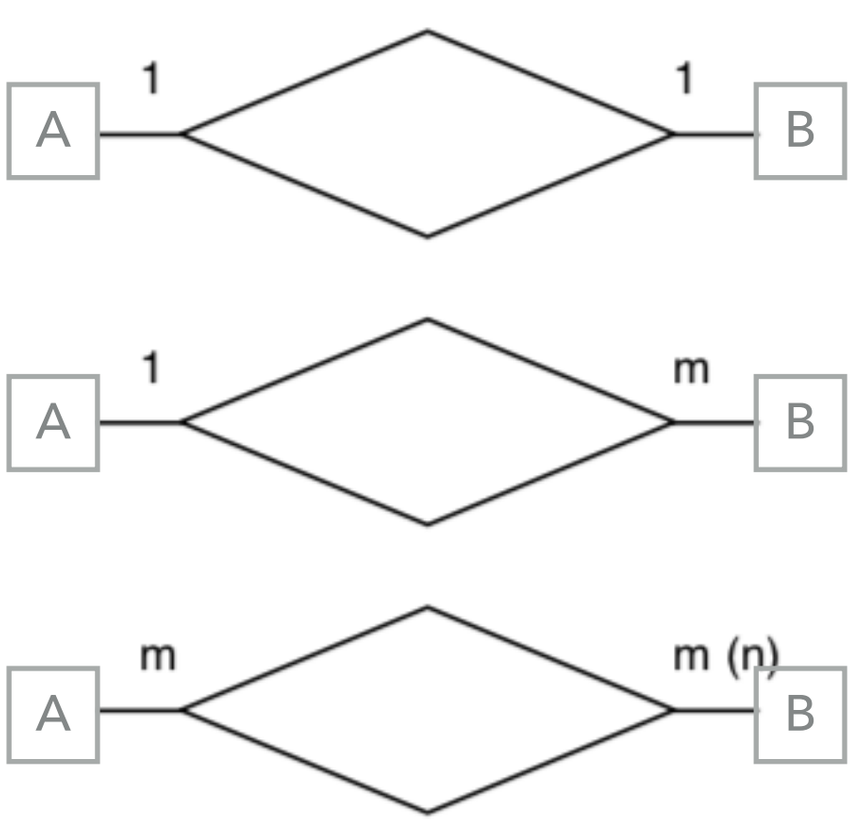
Das Entity-Relationship-Modell

- Zwischen den Klassen gibt es Beziehungen mit unterschiedlichen Merkmalen.
- Darstellung:



Das Entity-Relationship-Modell

- Beschreibung der Beziehung (Kardinalität) zwischen den Klassen



Jedem Objekt der Klasse A wird genau ein Objekt der Klasse B zugeordnet

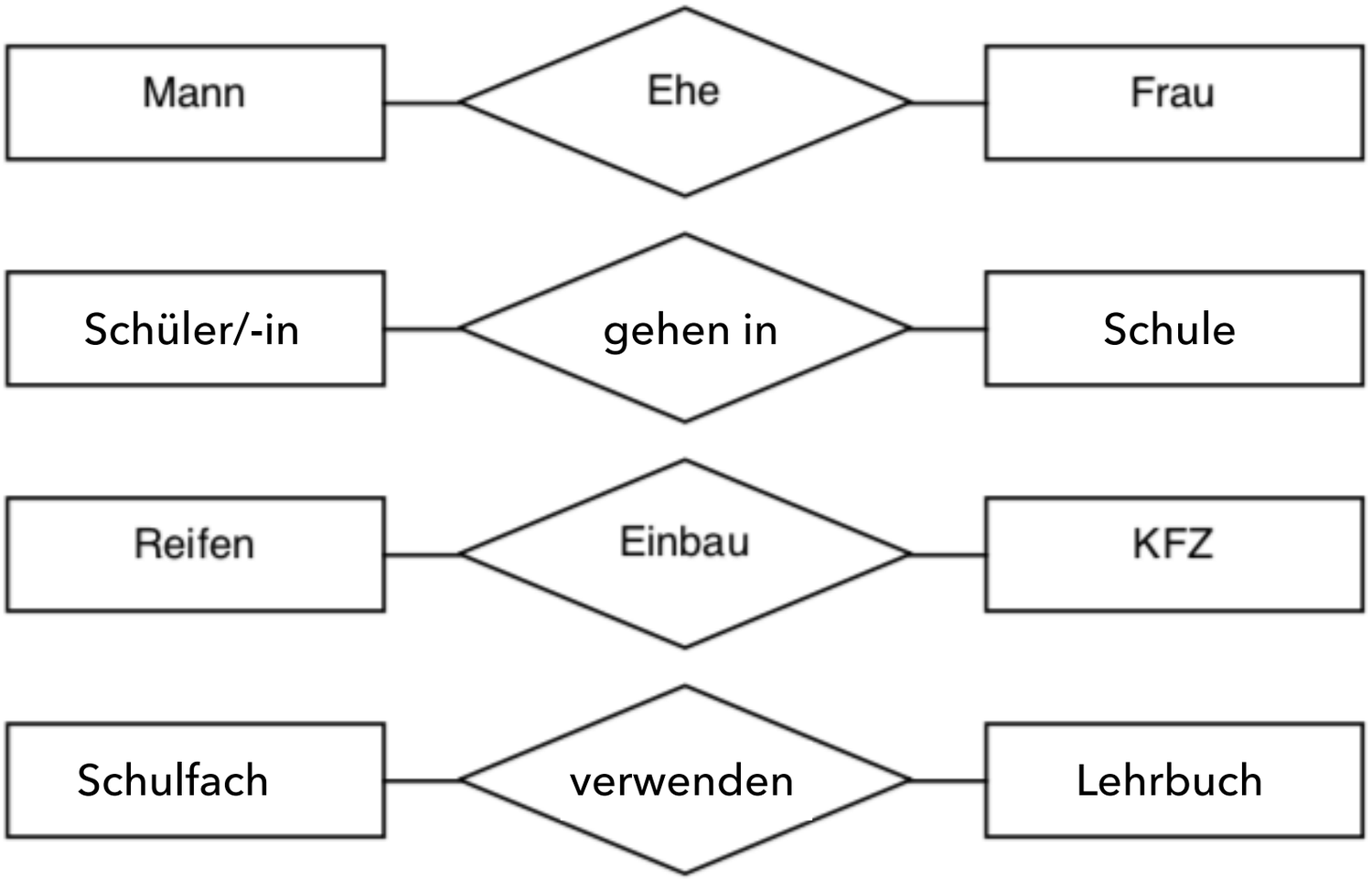
Jedem Objekt der Klasse A können mehrere Objekte der Klasse B zugeordnet werden. Umgekehrt wird aber jedem Objekt der Klasse B nur ein Objekt der Klasse A zugeordnet.

Jedem Objekt der Klasse A können mehrere Objekte der Klasse B zugeordnet werden und umgekehrt gilt dies auch.

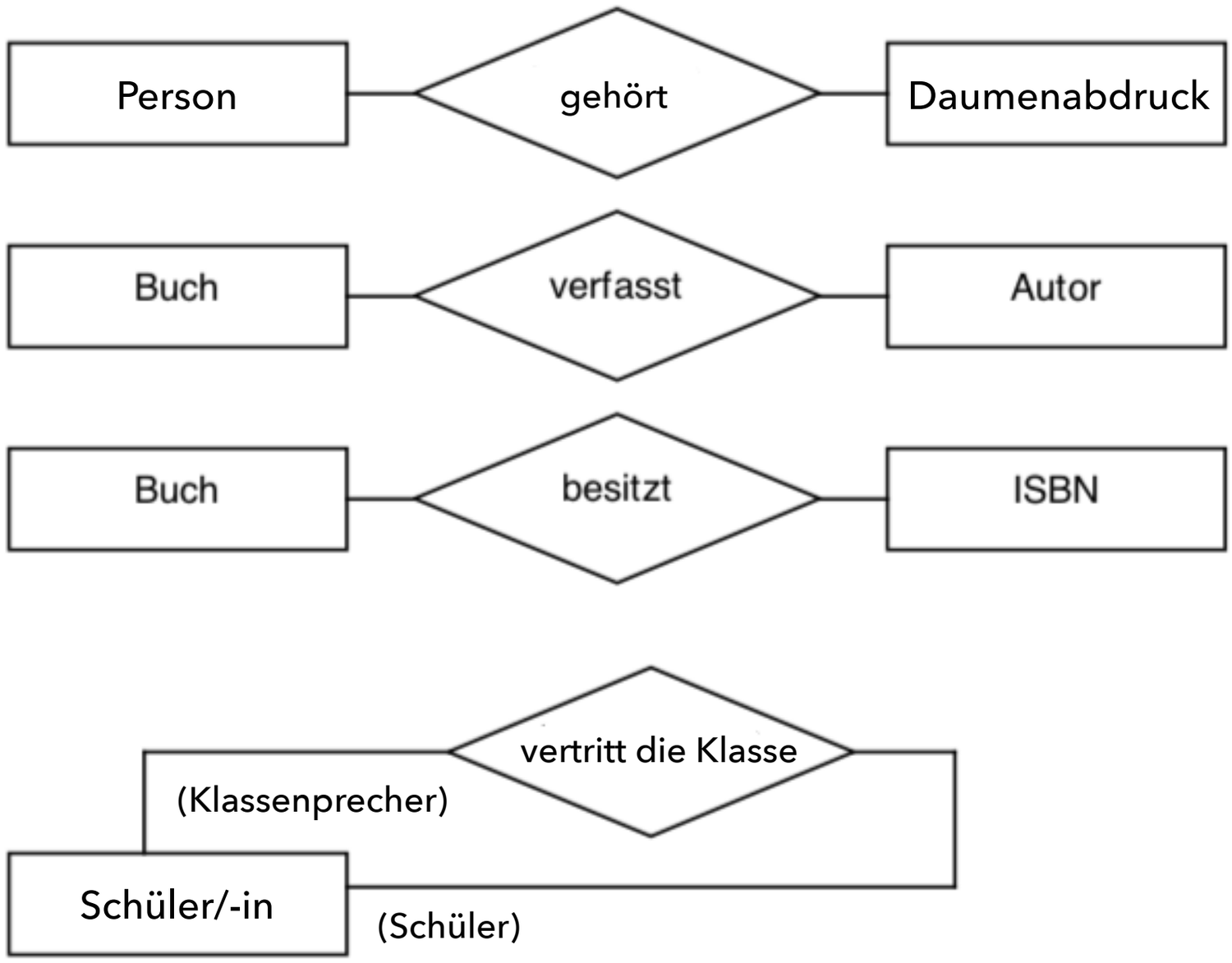
Beispiel

Beispiel

Übung


Mann
Ehe
Frau
Schüler/-in
gehen in
Schule
Reifen
Einbau
KFZ
Schulfach
verwenden
Lehrbuch

Übung


Person
gehört
Daumenabdruck
Buch
verfasst
Autor
Buch
besitzt
ISBN
vertritt die Klasse
(Klassenprecher)
Schüler/-in
(Schüler)

2.6 Abfragen relationaler Datenbanken

2.6.1 einfache Abfragen einer Tabelle

Die Abfragen erfolgen immer nach folgenden Schema:

Welche **Attribute**? Aus welcher **Tabelle**? Gibt es eine **Bedingung**? Soll es eine **Sortierung** geben?

Beispiel 1:

Wir möchten alle 28’er Fahrräder mit ihrer Nummer, ihren Modellnamen und Hersteller ausgeben.
Die Ausgabe soll aufsteigend sortiert nach dem Hersteller und anschließend dem Modell erfolgen.

1. Herausfinden von Tabelle(n), Attributen, Bedingung und Sortierung

   - Attribute: fnr, modell, hersteller
   - Tabelle: fahrraeder
   - Bedingung: rahmen = 28
   - Sortierung: modell - aufsteigend

   Hinweis zur Sortierung:
   aufsteigend (ASC; ascend, engl. hinaufsteigen)
   absteigend (DESC; descend, engl. hinabsteigen)

2.6.1 einfache Abfragen einer Tabelle

2. Wir sprechen mit der Datenbank per SQL (Structured Query Language)

- SELECT fnr, hersteller, modell FROM fahrraeder WHERE rahmen=28 ORDER BY hersteller ASC, modell ASC

Beispiel 2:

Wir möchten alle Kunden mit Kundennummer und Wohnort ausgeben.

1. Herausfinden von Tabelle(n), Attributen, Bedingung und Sortierung

    - Attribute: knr, name, vorname, ort
    - Tabelle: kunden
    - Bedingung: keine (alle Kunden)
    - Sortierung: keine

2. Wir sprechen mit der Datenbank per SQL

- SELECT knr, name, vorname, ort FROM kunden

Beispiel 2:

Wir möchten **alle** Attribute der Tabelle verleihdaten für den 4. Verleihvorgang.

1. Herausfinden von Tabelle(n), Attributen, Bedingung und Sortierung

    - Attribute: alle (*)
    - Tabelle: verleihdaten
    - Bedingung: verleihnr=4
    - Sortierung: keine

2. Wir sprechen mit der Datenbank per SQL

- SELECT * FROM verleihdaten WHERE verleihnr=4

2.6 Abfragen relationaler Datenbanken

2.6.2 Suchbedingungen mit „WHERE“

Oft werden Namen oder Begriffe in der Datenbank gesucht.

Suche nach einem Namen oder Begriff:

WHERE name= ' Schulze '

WHERE modell= ' Orkan '

Wenn nur ein Teil des Namens oder Begriffes bekannt ist kann ein Platzhalter verwendet werden.

WHERE name like ' Ad%‘ ' - Namen beginnen mit Ad…

WHERE name like ' %lein‘ ' - Namen enden auf …lein

Beispiel:

- SELECT knr, name, vorname, ort FROM kunden WHERE name like ‚%lein'
- SELECT knr, name, vorname, ort FROM kunden WHERE name like 'W%lein'

2.6 Abfragen relationaler Datenbanken

2.6.2 Weitere Suchbedingungen mit „WHERE“

Unbestimmte Bedingungen mit Größer und Kleiner für Zahlen oder dem Datum

Bsp.:

- Alle Fahrräder deren Rahmen kleiner als 27“ ist.

```
select * from fahrraeder where rahmen < 27
```

- Alle Fahrräder deren Anschaffung ab dem (größergleich) 01.10.2017 liegt.

```
select * from fahrraeder where anschaffungsdatum >=  ' 2017-10-01 ' .
```

Übungen

Vor der Formulierung der SQL-Anweisung vervollständige die folgenden Angaben:

- Attribute:
- Tabelle:
- Bedingung:
- Sortierung:

1. Zeige alle Kunden mit Kundennummer, Name, Vorname und Ort an, welche in Köln wohnen. Sortiere die Kunden nach Namen, Vornamen.

2. Suche den Lieferanten Mirage… (beginnt mit „Mirage“). Es soll der Vorname, Name, Strasse, PLZ und Ort ausgegeben werden.

3. Zeige alle Fahrräder des Herstellers „Hirsch“ an. Ausgabe von Fahrradnummer, Modell, Rahmengröße. Sortiert nach Rahmengröße absteigend (DESC)

Übung zur Abfrage einer Tabelle (Filtern, Sortieren)

- Erstelle eine Ausgabe aller Fahrräder mit einer Rahmengröße von 27 Zoll und kleiner! Die Ausgabe enthält die Attribute fnr, hersteller, modell, rahmen und ist nach Rahmengröße absteigend und nach Hersteller und Modell aufsteigend sortiert.

- Erstelle eine Liste aller Fahrräder, die am, 07.03.2017 angeschafft wurden. Sortiere nach fnr und gibt die Attribute fnr, Hersteller, Modell, Sonderzubehör aus.

- Suche nach einem Kunden aus Gummers…? Gib alle Attribute der Tabelle Kunden aus.

2.6.3 Abfragen aus mehreren Tabellen (Beziehung)

Die Daten einer Datenbank sind auf mehrere Tabellen aufgeteilt, damit mehrfache (Redundanzen) oder ungenaue (Anomalien) Einträge vermieden werden. Für bestimmte Abfragen müssen die verteilt gespeicherten Daten allerdings wieder zusammengeführt werden. Verwendung findet hier die Verbindung zwischen Fremd- und Primärschlüssel.

Beispiel:
Ausgabe aller Millionenstädte mit ihren Namen, ihrer Einwohnerzahl und dem Land in denen sie sich befinden. Absteigend sortiert nach der Einwohnerzahl.

STADT

Fremdschlüssel

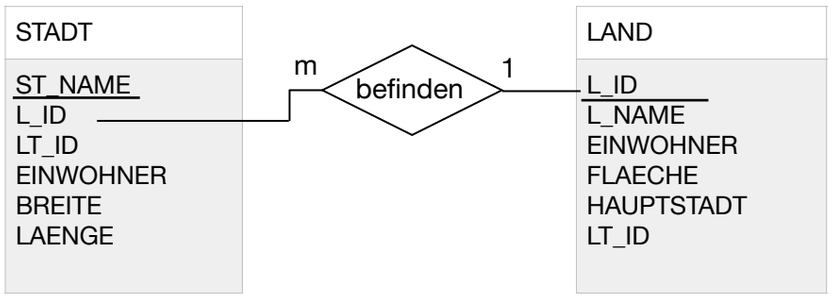
ST_NAME	L_ID	LT_ID	EINWOHNER	BREITE	LAENGE
Aachen	D	NRW	...	...	...
Aalborg	DK	DK	...	...	...

LAND

Primärschlüssel

L_ID	L_NAME	EINWOHNER	FLAECHE	HAUPTSTADT	LT_ID
D	Deutschland	...	...	Berlin	B
DK	Daenemark	...	...	Kopenhagen	DK

2.6.3 Abfragen aus mehreren Tabellen (Beziehung)



ausführliche Angabe des Attributes (Tabelle.Spalte)

- Attribut(e): STADT.ST_NAME, STADT.EINWOHNER, LAND.L_NAME
- Tabelle(n): STADT, LAND
- Bedingung(en): STADT.EINWOHNER>1000000
- Sortierung(en): STADT.EINWOHNER DESC

Verknüpfung (INNER JOIN) der Schlüsselattribute:

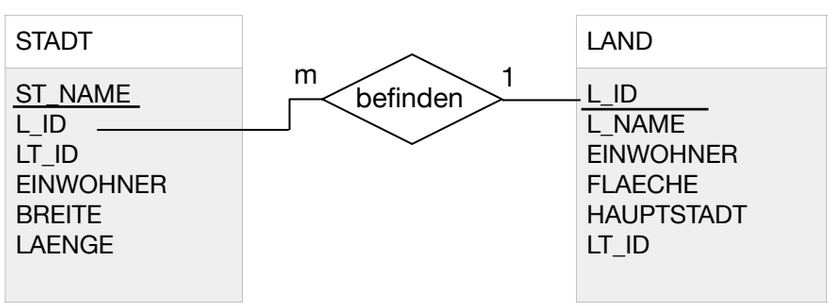
Primärschlüssel: LAND.L_ID

Fremdschlüssel: STADT.L_ID

Wir sprechen SQL:

```
SELECT    STADT.ST_NAME, STADT.EINWOHNER, LAND.L_NAME
FROM   STADT   INNER JOIN    LAND    ON    STADT.L_ID=LAND.L_ID
WHERE STADT.EINWOHNER>1000000
ORDER BY  STADT.EINWOHNER DESC
```

2.6.3 Abfragen aus mehreren Tabellen (Beziehung)



ausführliche Angabe des Attributes (Tabelle.Spalte)

- Attribute: STADT.ST_NAME, STADT.EINWOHNER, LAND.L_NAME
- Tabelle(n): STADT, LAND
- Bedingung: STADT.EINWOHNER>1000000
- Sortierung: STADT.EINWOHNER DESC

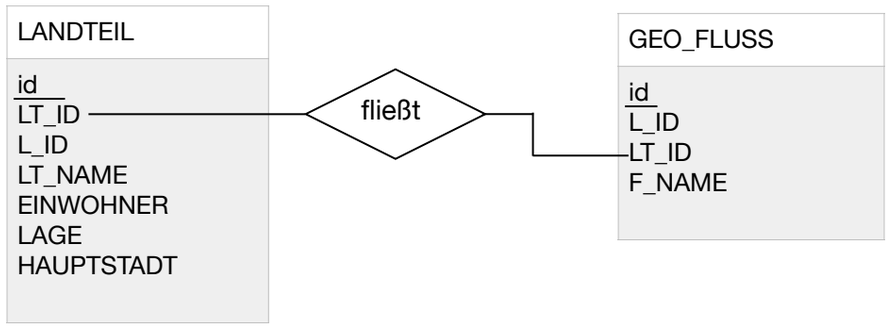
Verknüpfung (INNER JOIN) der Schlüsselattribute:
Primärschlüssel: LAND.L_ID
Fremdschlüssel: STADT.L_ID

Wir sprechen SQL:

```
SELECT STADT.ST_NAME, STADT.EINWOHNER, LAND.L_NAME FROM STADT INNER JOIN LAND ON STADT.L_ID =
LAND.L_ID WHERE STADT.EINWOHNER>1000000 ORDER BY STADT.EINWOHNER DESC
```

2.6.3 Abfragen aus mehreren Tabellen (Beziehung)

Beispiel:
Ausgabe aller Landesteile, durch die die Donau fließt, sortiert nach den Namen der Landesteile.



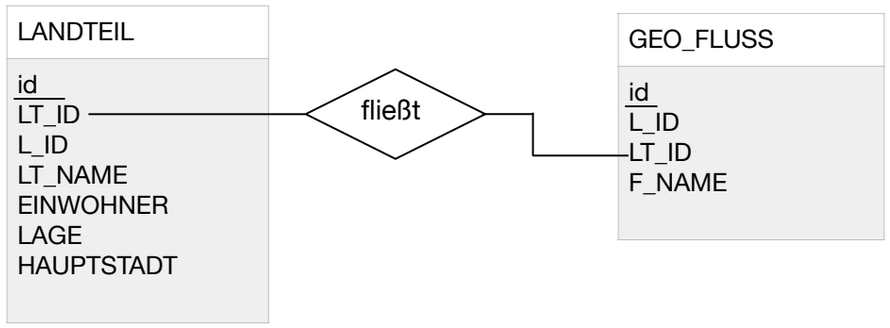
- Attribute:
- Tabelle(n):
- Bedingung:
- Sortierung:

Verknüpfung (INNER JOIN) der Schlüsselattribute:
Primärschlüssel:
Fremdschlüssel:

Wir sprechen SQL:

2.6.3 Abfragen aus mehreren Tabellen (Beziehung)

Beispiel:
Ausgabe aller Landesteile, durch die die Donau fließt, sortiert nach den Namen der Landesteile.



- Attribute: LANDTEIL.LT_NAME
- Tabelle(n): GEO_FLUSS, LANDTEIL
- Bedingung: F_NAME="Donau"
- Sortierung: LANDTEIL.LT_NAME

Verknüpfung (INNER JOIN) der Schlüsselattribute:
Primärschlüssel: LANDTEIL.LT_ID
Fremdschlüssel: GEO_FLUSS.LT_ID

Wir sprechen SQL:

```
SELECT LANDTEIL.LT_NAME FROM GEO_FLUSS INNER JOIN LANDTEIL ON GEO_FLUSS.LT_ID = LANDTEIL.LT_ID
WHERE F_NAME="Donau" ORDER BY LANDTEIL.LT_NAME
```