

- Zwischen den Relationen/Tabellen der Geographie-Datenbank bestehen Beziehungen
- Beispiel: Tabellen **STADT** und **BUNDESLAND**
  - Jede Stadt liegt in **genau einem** Bundesland
  - Die meisten Bundesländer beinhalten **mehrere** Städte
- Diese Art von Beziehung wird als 1:n-Beziehung bezeichnet
  - Wie wird sie in der Datenbank gespeichert?

- Die n-Seite wird um einen Verweis auf die 1-Seite erweitert
  - Dieser Verweis muss eine eindeutige Zuordnung erlauben  $\Rightarrow$  Verweis auf den Schlüssel der 1-Seite
  - Man nennt diesen Verweis daher **Fremdschlüssel**
- Im Beispiel:
  - Jede Stadt verfügt über ein Datenfeld **bkuerzel**

- Beispiel: Tabellen **FLUSS** und **BUNDESLAND**
- Warum ist es hier nicht sinnvoll, eine der beteiligten Tabellen um einen Fremdschlüssel zu erweitern?
  - Da jedes Element der einen Seite mit beliebig vielen Elementen der anderen Seiten in Beziehung stehen kann, reicht ein einzelner Fremdschlüssel nicht aus
- Stattdessen: Verwendung einer **Beziehungstabelle**
  - Die Beziehungstabelle referenziert mit jeder Zeile zwei Elemente, die miteinander in Beziehung stehen

# Beispiel Beziehungstabelle

FLUSS:

id	name
1	Rhein
2	Donau
3	Neckar
...	...

BUNDESLAND:

id	name
1	Baden-Württemberg
2	Bayern
3	Rheinland-Pfalz
...	...

Beziehungstabelle BL\_FLUSS:

f_id	b_id
1	1
1	3
2	1
2	2
3	1
...	...

# Abfragen über mehrere Tabellen (1)

Beispieltabellen:

VATER:

v_id	name
1	Karl
2	Gustav

KIND:

k_id	name	v_id
1	Lilly	1
2	Lara	1
3	Lucie	2

- Welches Ergebnis liefert die Abfrage

**select** \* **from** VATER, KIND?

- Die Abfrage bildet das **kartesische Produkt**  $VATER \times KIND$  über die beiden Tabellen, d. h. jeder Datensatz aus der Tabelle **VATER** wird mit jedem Datensatz aus der Tabelle **KIND** kombiniert

## Abfragen über mehrere Tabellen (2)

VATER:

v_id	name
1	Karl
2	Gustav

KIND:

k_id	name	v_id
1	Lilly	1
2	Lara	1
3	Lucie	2

VATER  $\times$  KIND:

v_id	name	k_id	name	v_id
1	Karl	1	Lilly	1
1	Karl	2	Lara	1
1	Karl	3	Lucie	2
2	Gustav	1	Lilly	1
2	Gustav	2	Lara	1
2	Gustav	3	Lucie	2

## Abfragen über mehrere Tabellen (3)

VATER \*KIND:

v_id	name	k_id	name	v_id
1	Karl	1	Lilly	1
1	Karl	2	Lara	1
1	Karl	3	Lucie	2
2	Gustav	1	Lilly	1
2	Gustav	2	Lara	1
2	Gustav	3	Lucie	2

- Welcher dieser Kombinationen sind sinnvoll?
  - Nur die, bei denen Schlüssel und Fremdschlüssel übereinstimmen

## Abfragen über mehrere Tabellen (4)

- Wir müssen die SQL-Abfrage so formulieren, dass ungültige Kombinationen nicht Teil des Ergebnisses sind:

- Möglichkeit 1

```
select * from VATER, KIND where VATER.v_id = KIND.v_id
```

- Möglichkeit 2 (zu bevorzugen!)

```
select * from VATER join KIND on VATER.v_id = KIND.v_id
```

- Die hier durchgeführte Datenbankoperation nennt sich **(inner) Join**



- Öffne die Datei “geographie.odt” mit LibreOffice Base
- Ermittle folgende Daten mittels geeigneter SQL-Abfragen:
  - Name und Einwohnerzahl aller Städte, die in Bayern oder Baden-Württemberg liegen, in alphabetischer Reihenfolge (die **bkuerzel** von Bayern oder Baden-Württemberg dürfen **nicht** als bekannt vorausgesetzt werden)
  - Die Namen aller Flüsse, die durch Baden-Württemberg fließen

## Aufgabe 2

- Öffne die Datei “sportverein.odt” mit LibreOffice Base
- Ermittle folgende Daten mittels geeigneter SQL-Abfragen:
  - Der Name aller Sportarten, die der Verein anbietet
  - Alle Mitglieder mit vollständiger Adresse (also
  - einschließlich PLZ und Ortsname)
  - Vor- und Nachname aller Trainer
  - Die Gesamtzahl aller Mitglieder
  - Alle Mitglieder, die in Turing wohnen
  - Alle Sportarten mit ihren jeweiligen Trainern
  - Alle Sportarten, deren Trainer mindestens 150 Euro erhält
  - Alle Fußballspieler aus Spalting oder Infohausen

- Allgemein:

```
INSERT INTO <Tabelle> (<Datenfelder>) VALUES (<Werteliste>)
```

- Beispiel:

```
INSERT INTO KIND VALUES (4, "Leon", 2)
```

```
INSERT INTO KIND (kid, name) VALUES (5, "Luca")
```

- Allgemein:

```
UPDATE <Tabelle> SET <Datenfeld>=<Wert> WHERE <Bedingung>
```

- Beispiel:

```
UPDATE VATER SET name="Theodor" WHERE name="Karl"
```

- Bemerkung: Wie bei **SELECT** ist die Bedingung optional. Fehlt sie, werden jedoch **alle** Datensätze aktualisiert!

- Allgemein:

```
DELETE FROM <Tabelle> WHERE <Bedingung>
```