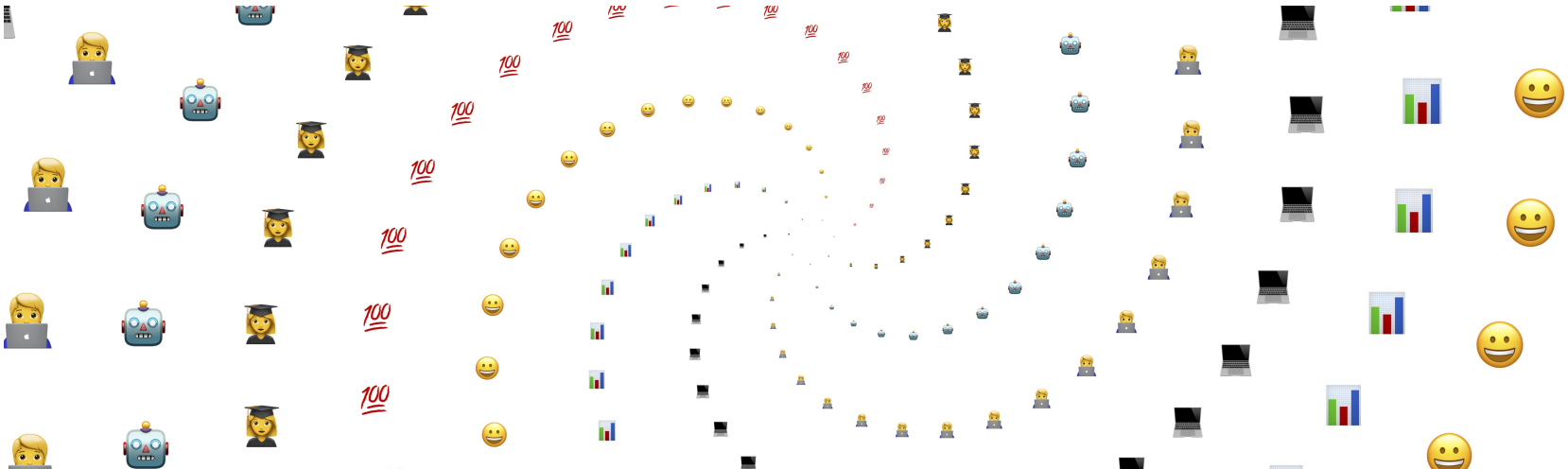
Datenbanken

👩‍💻 Datenbanken und SQL

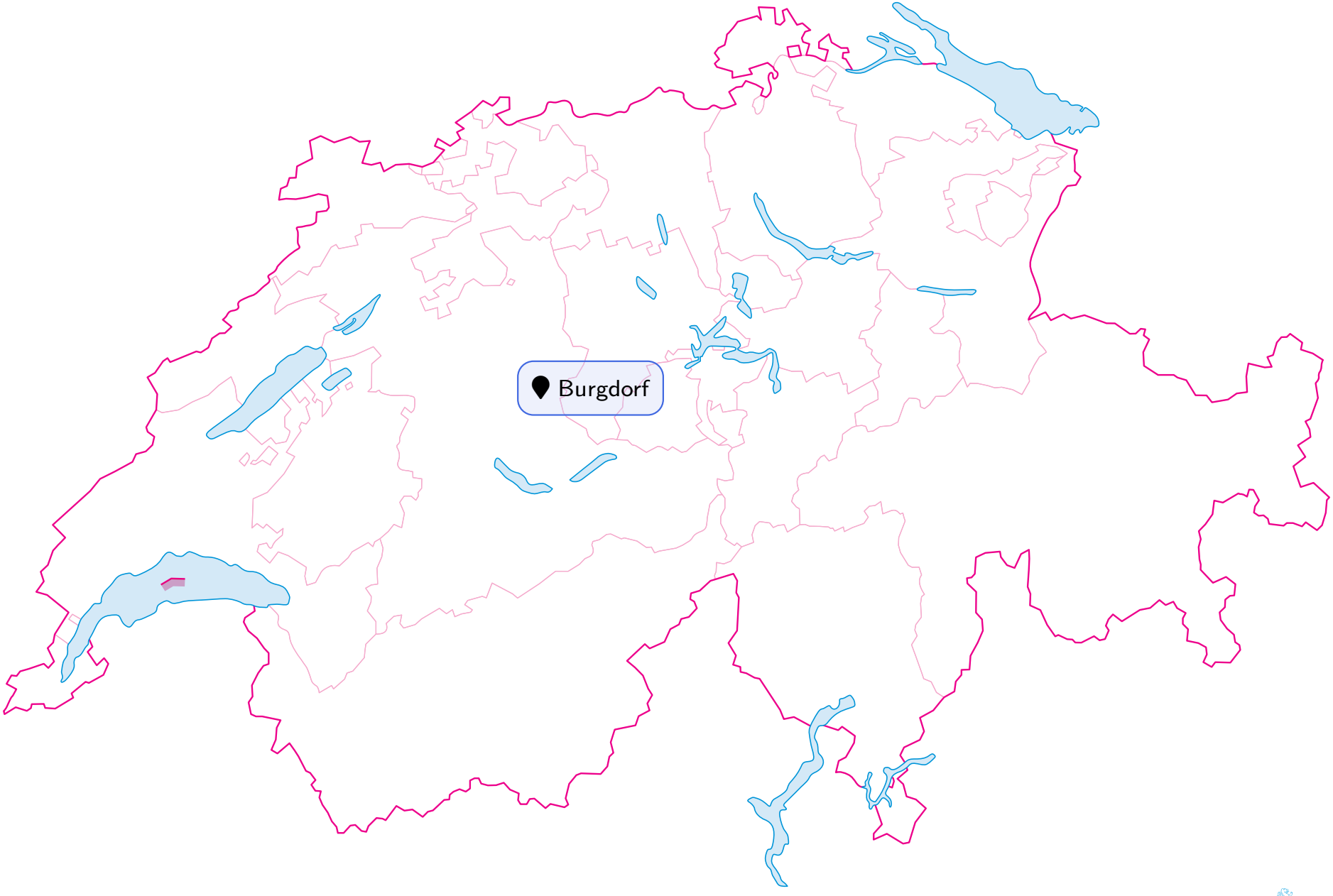
Naoki Peter[1] Cyril Wendl [2]

[1]Fachschaft Informatik
Kantonsschule Zürich-Nord

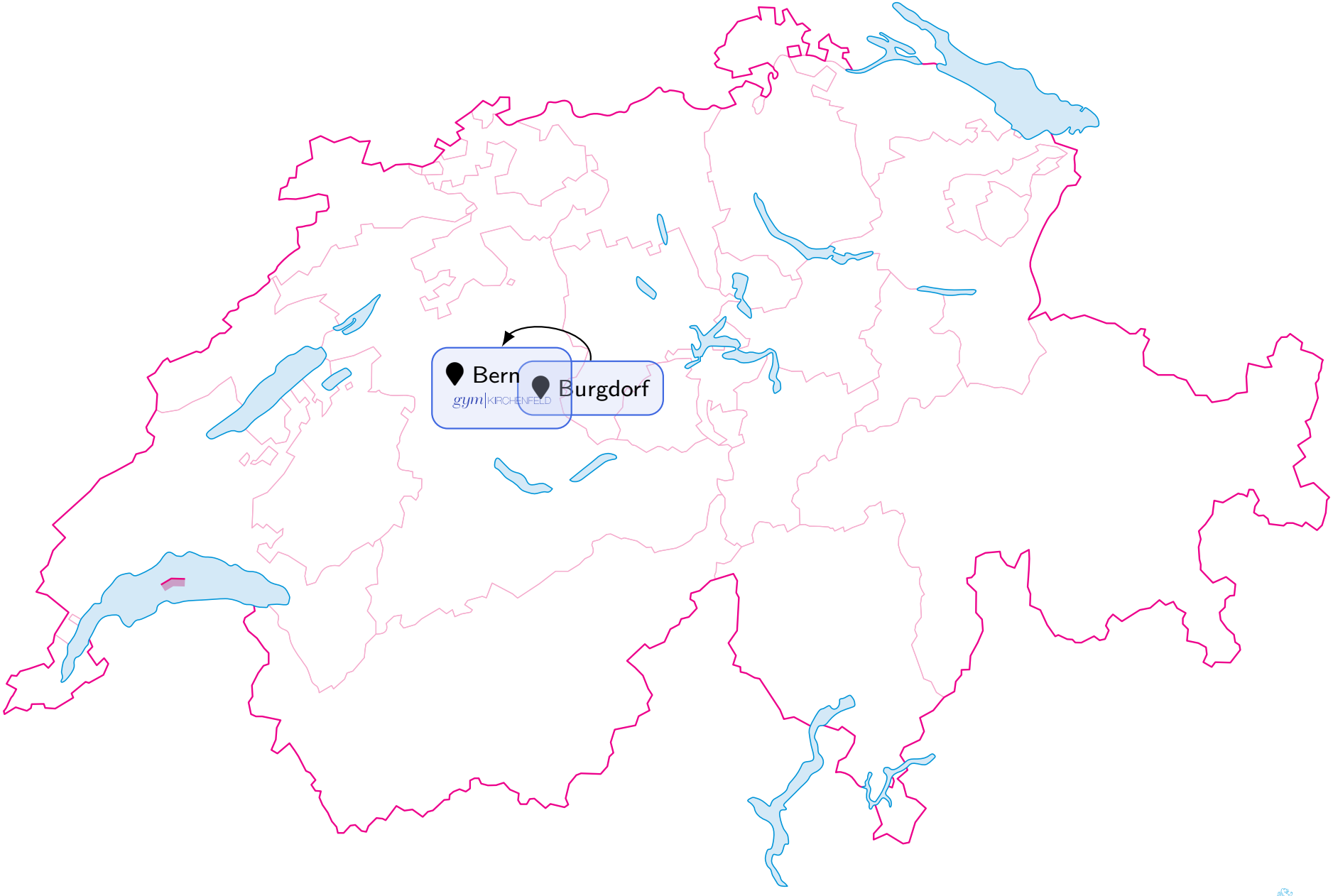
[2]Fachschaft Informatik
Kantonsschule im Lee



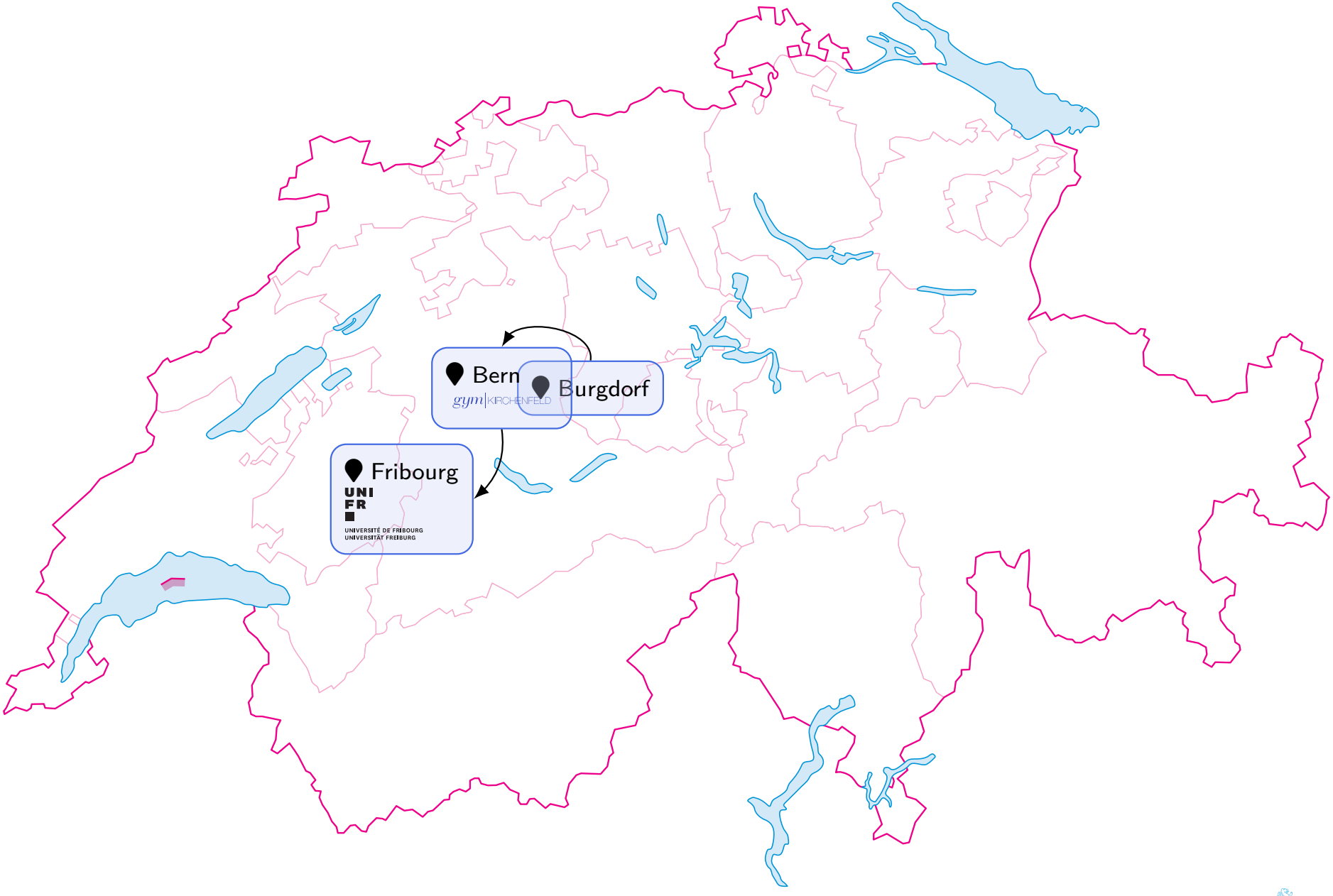
Vorstellung



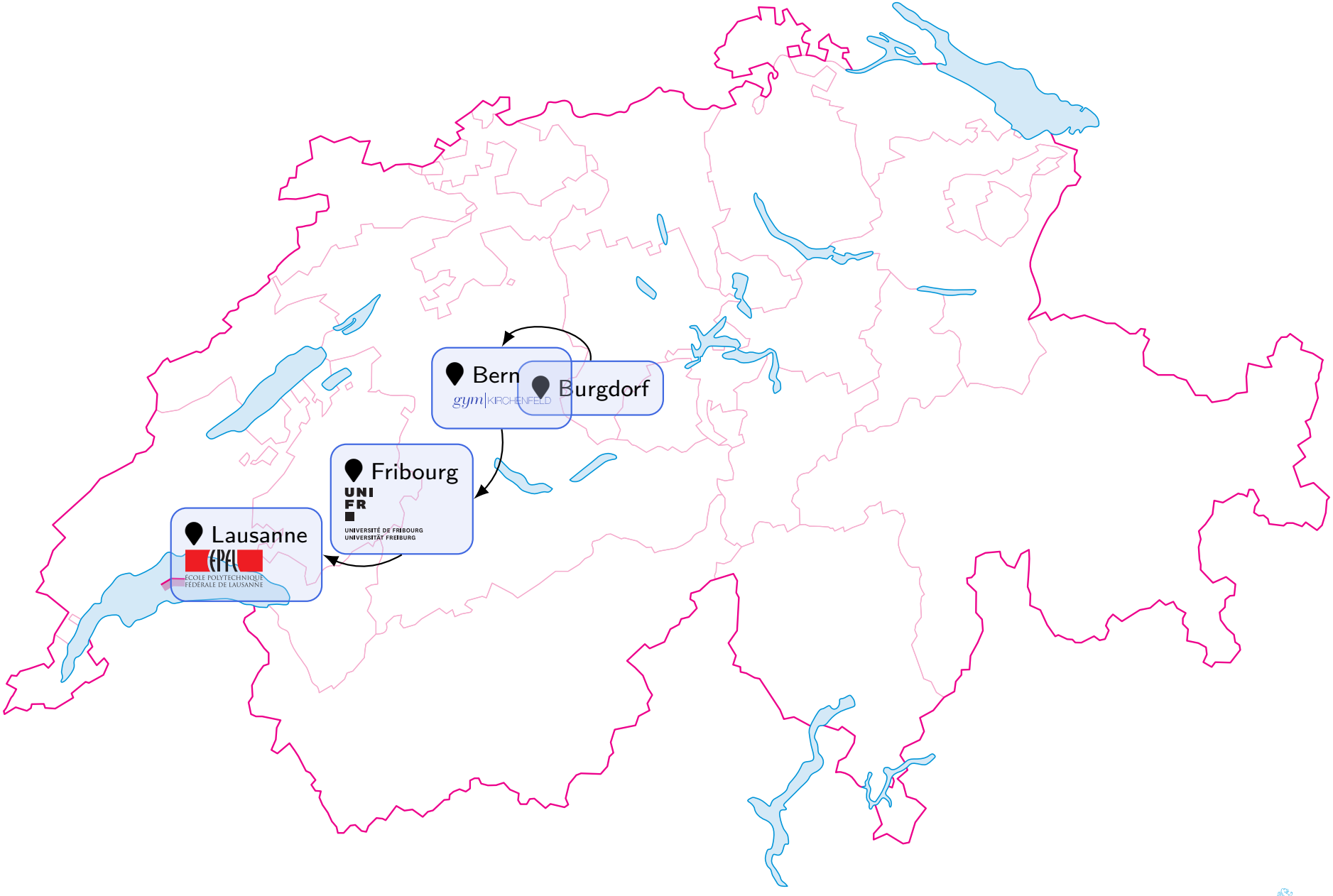
Vorstellung



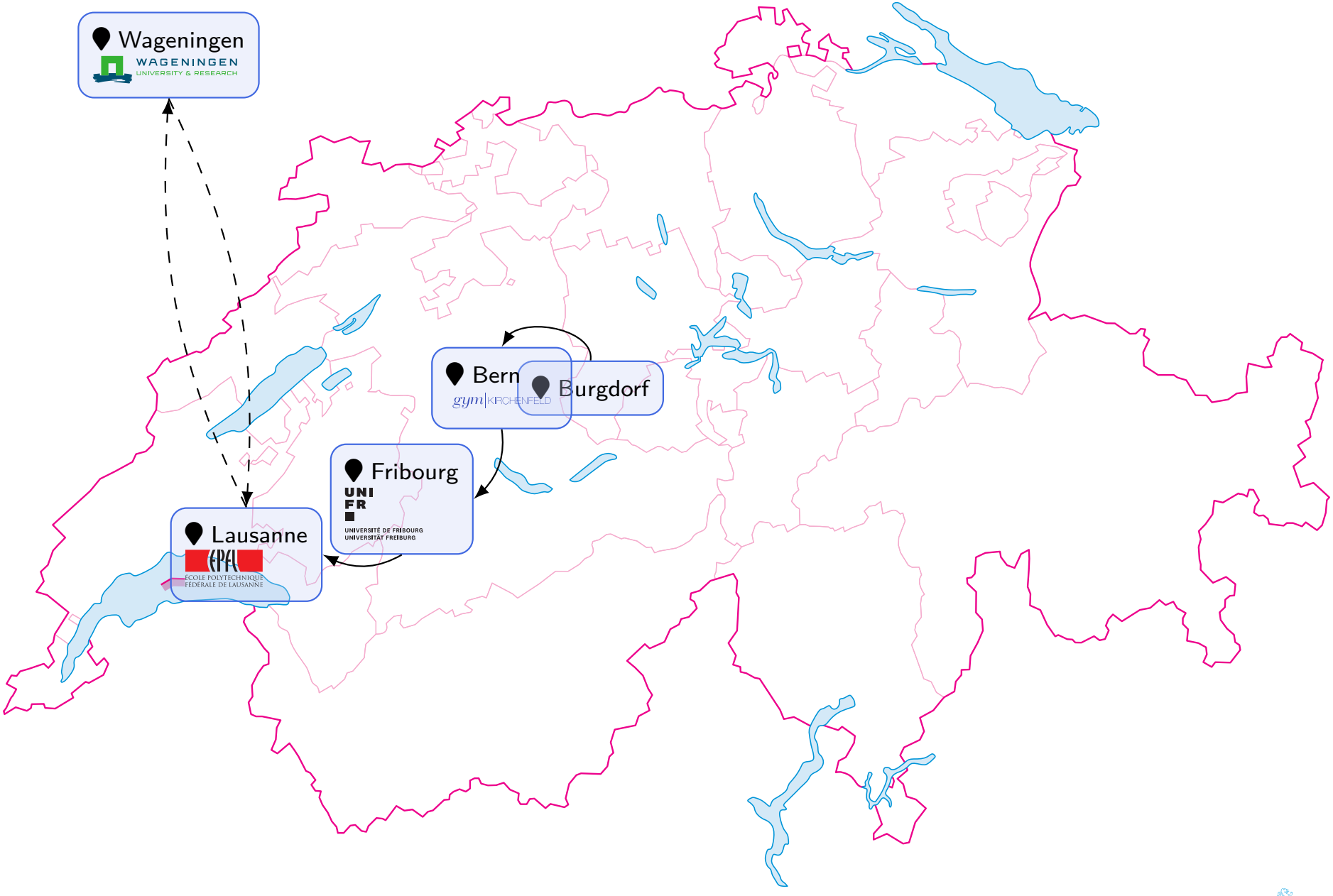
Vorstellung



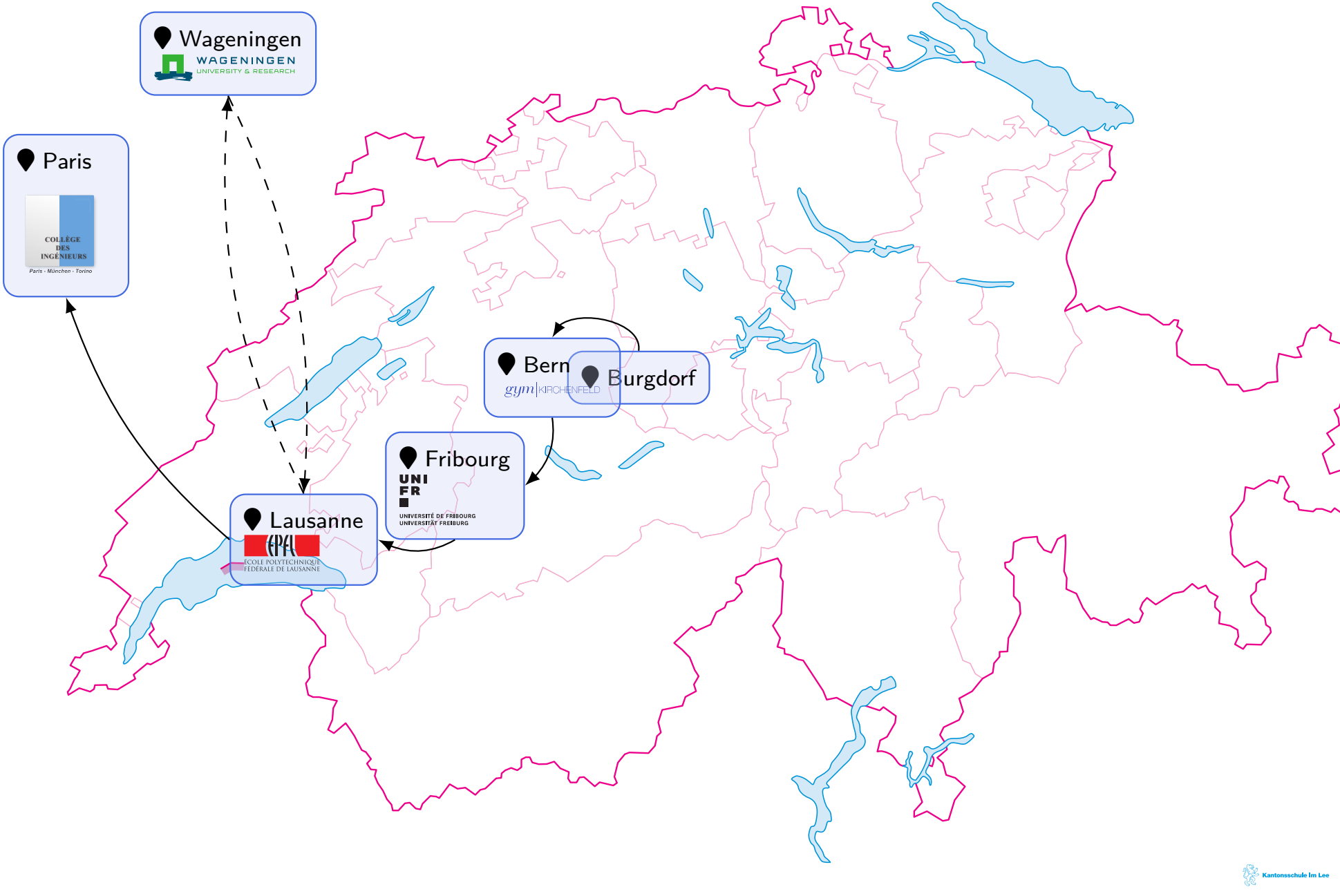
Vorstellung



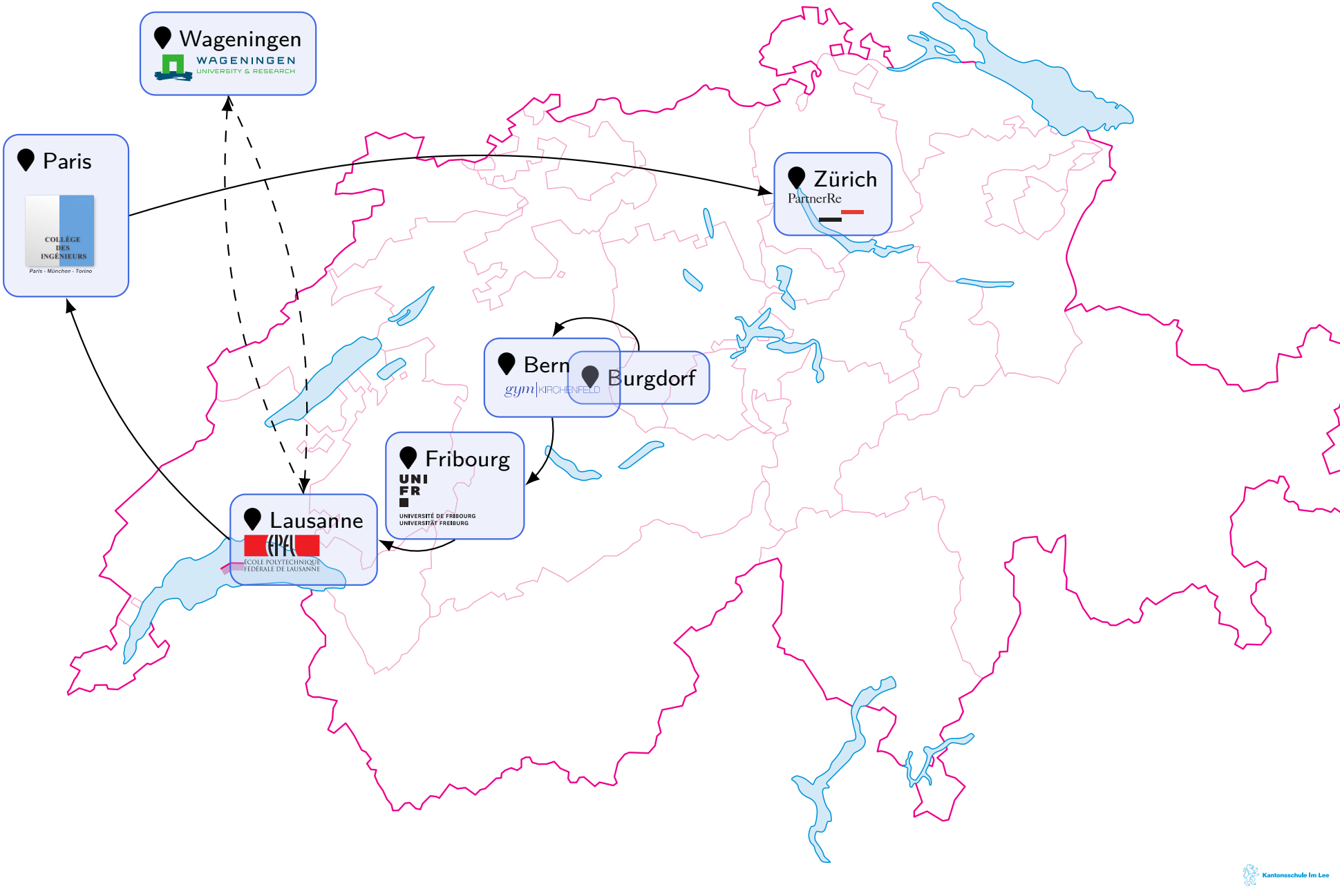
Vorstellung



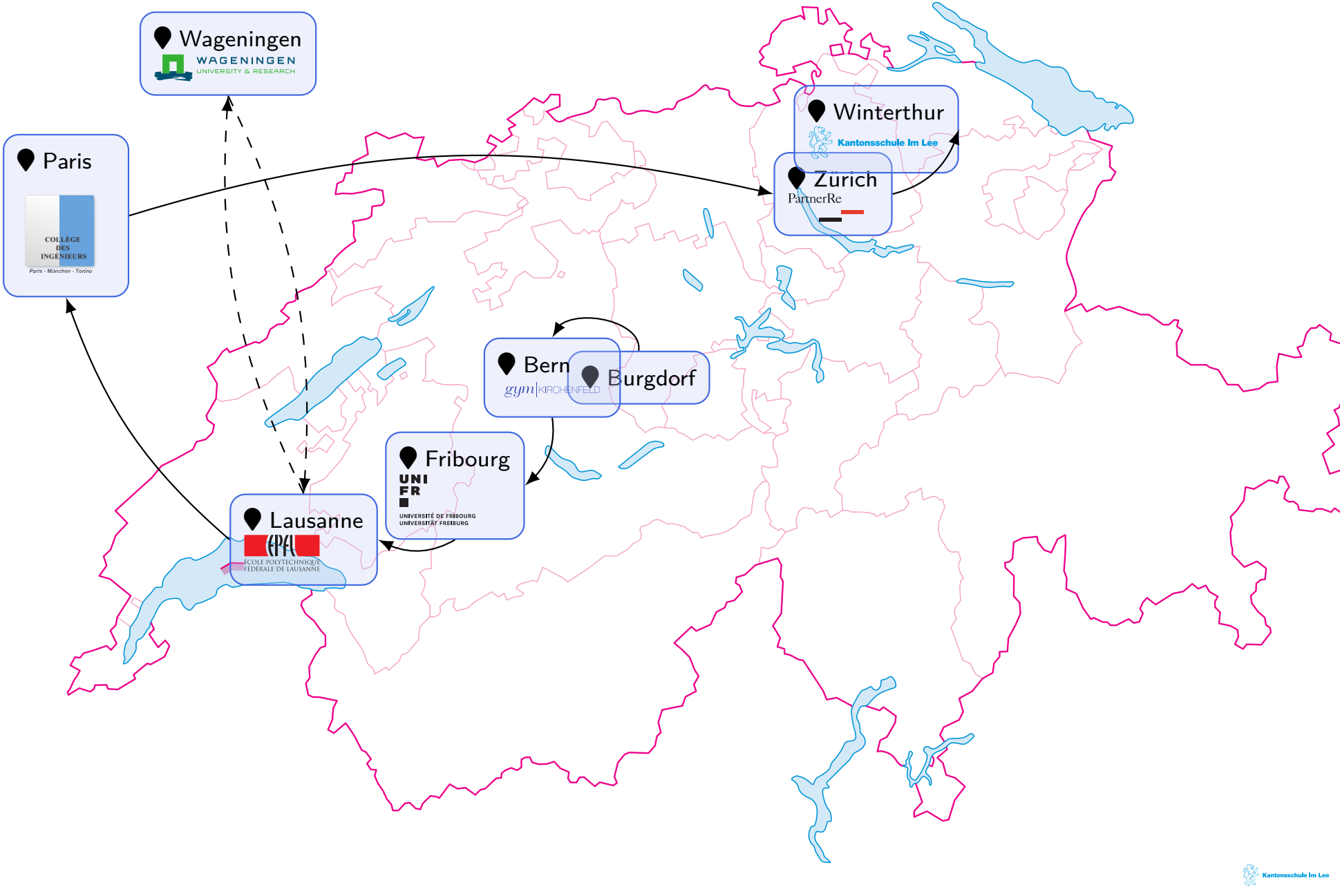
Vorstellung



Vorstellung



Vorstellung



Interessen

wendl.ch

Hauptziel

Abschluss: Maturität

Hauptziel

~~***Abschluss:*** *Maturität*~~

Spass an der Informatik und am eigenen Computer gewinnen!

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“
  - „Aktuell findet an der KLW Unterricht statt.“

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“
  - „Aktuell findet an der KLW Unterricht statt.“
  - „Das Billett von Zürich nach Winterthur kostet CHF 6.80.“

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“
  - „Aktuell findet an der KLW Unterricht statt.“
  - „Das Billett von Zürich nach Winterthur kostet CHF 6.80.“
  - „Das Logo der KLW besteht aus blauen und roten Farben.“

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“
  - „Aktuell findet an der KLW Unterricht statt.“
  - „Das Billett von Zürich nach Winterthur kostet CHF 6.80.“
  - „Das Logo der KLW besteht aus blauen und roten Farben.“
  - ...

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“
  - „Aktuell findet an der KLW Unterricht statt.“
  - „Das Billett von Zürich nach Winterthur kostet CHF 6.80.“
  - „Das Logo der KLW besteht aus blauen und roten Farben.“
  - ...
- In der **Informatik**: irgendetwas mit Nullen und Einsen
  Beispiel:

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“
  - „Aktuell findet an der KLW Unterricht statt.“
  - „Das Billett von Zürich nach Winterthur kostet CHF 6.80.“
  - „Das Logo der KLW besteht aus blauen und roten Farben.“
  - ...
- In der **Informatik**: irgendetwas mit Nullen und Einsen
  Beispiel:
  - ...10100010101111011101001001...

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“
  - „Aktuell findet an der KLW Unterricht statt.“
  - „Das Billett von Zürich nach Winterthur kostet CHF 6.80.“
  - „Das Logo der KLW besteht aus blauen und roten Farben.“
  - ...
- In der **Informatik**: irgendetwas mit Nullen und Einsen
  Beispiel:
  - ...10100010101111011101001001...
  - ...11101110110000111011010111...

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“
  - „Aktuell findet an der KLW Unterricht statt.“
  - „Das Billett von Zürich nach Winterthur kostet CHF 6.80.“
  - „Das Logo der KLW besteht aus blauen und roten Farben.“
  - ...
- In der **Informatik**: irgendetwas mit Nullen und Einsen
  Beispiel:
  - ...10100010101111011101001001...
  - ...11101110110000111011010111...
  - ...

Was sind *Daten*?

- **„Daten“** = Plural von *Datum*
- *Datum* = Fakt(um) (von lateinisch *datum* = gegeben, *dare* = geben), also etwas **Gegebenes**
  Beispiel:
  - „Mein Nachbar heisst Marco Odermatt.“
  - „Aktuell findet an der KLW Unterricht statt.“
  - „Das Billett von Zürich nach Winterthur kostet CHF 6.80.“
  - „Das Logo der KLW besteht aus blauen und roten Farben.“
  - ...
- In der **Informatik**: irgendetwas mit Nullen und Einsen
  Beispiel:
  - ...10100010101111011101001001...
  - ...11101110110000111011010111...
  - ...
- Welche Codierung welchem *Datum* entspricht, bestimmt der / die ProgrammiererIn

Was sind *Daten*?

Daten sind...

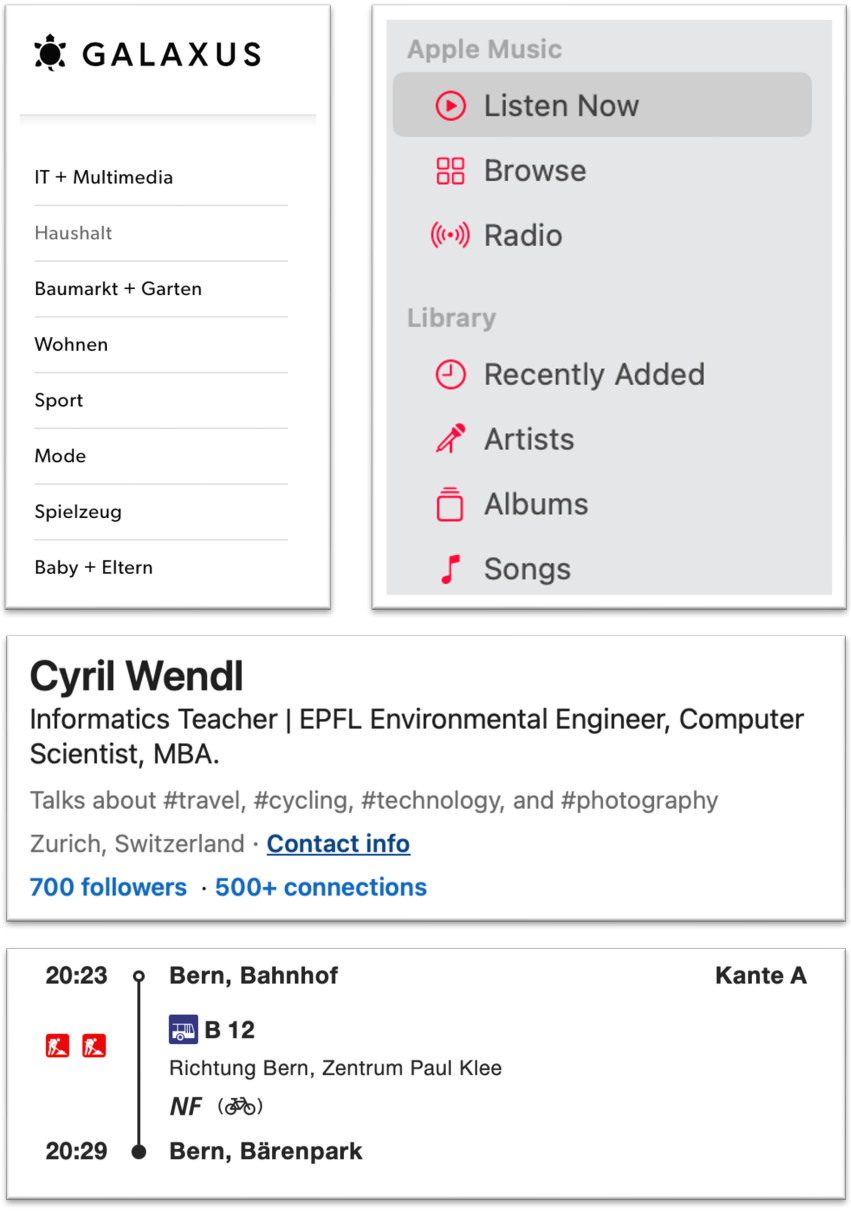
... am häufigsten in Firmen, Verwaltungen, Unternehmen, Schulen, und vielen anderen Bereichen des täglichen Lebens: **Datenbanken**, bzw. **Tabellen**

Wo(zu) braucht es Datenbanken?

Daten sind omnipräsent...

- **Social Media**: Instagram, TikTok, LinkedIn, etc...
- **Shopping**: Galaxus, AliExpress, etc...
- **Netzwerke**: SBB, swissgrid (Strom-Netzwerk), etc...

Wie können wir mit den riesigen Datenmengen umgehen und sinnvolle Einsichten daraus gewinnen?

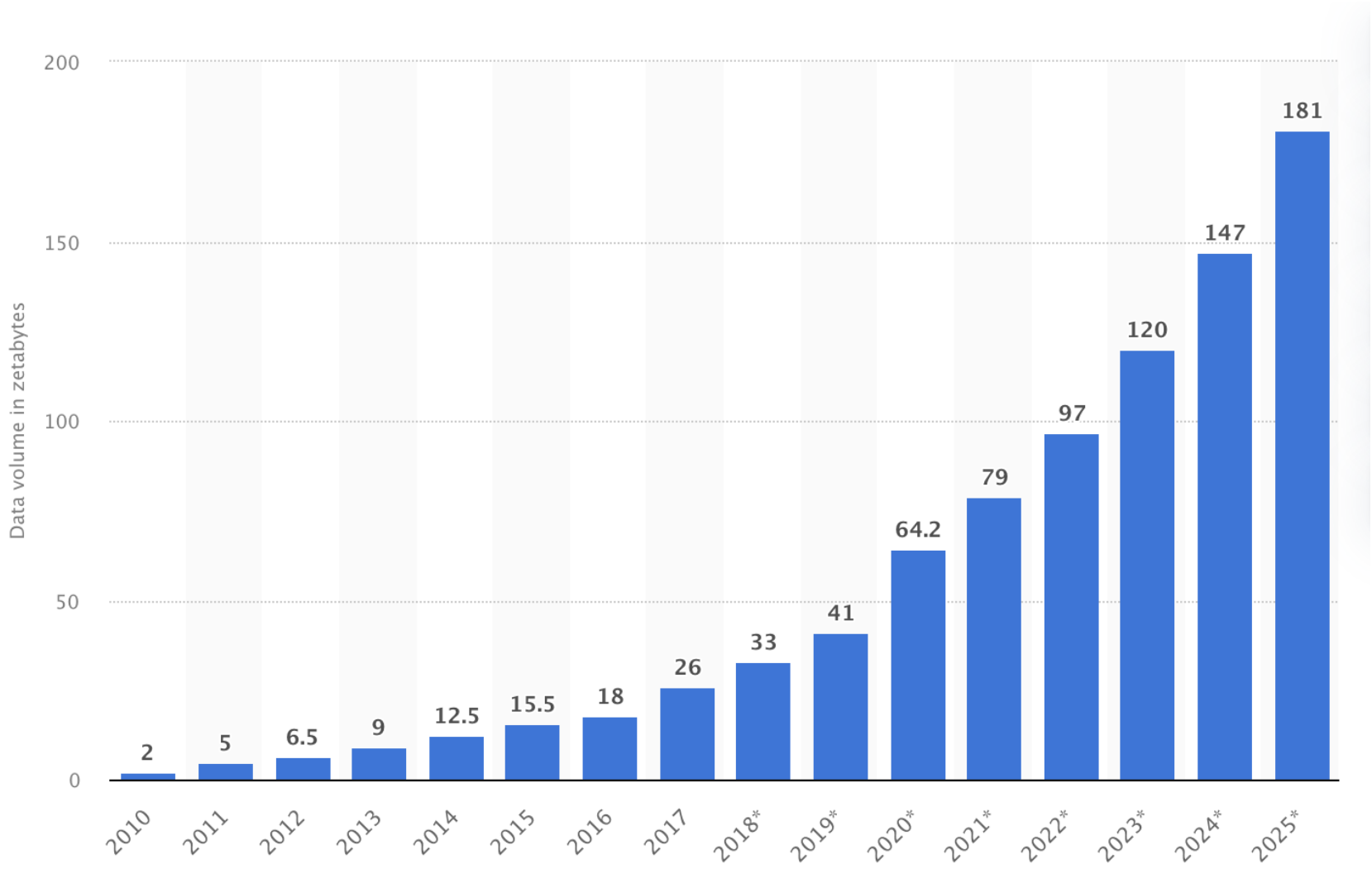


Datenmengen und Speicherplatz

Ein **B**yte = 8 **bit**

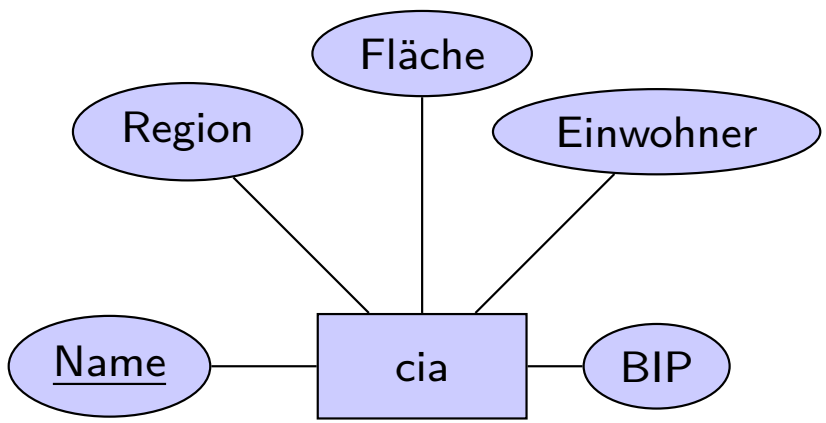
Masseinheit	**Dezimalsystem**		Grössenordnung
KB (Kilobyte)	10^3 B(yte)	1'000 B	Eine Text-Datei
MB (Megabyte)	10^6 B	1'000'000 B	Eine Musik-Datei
GB (Gigabyte)	10^9 B	1'000'000'000 B	Eine Video-Datei
TB (Terabyte)	10^{12} B	1'000'000'000'000 B	Kleiner Firmen-Server
PB (Petabyte)	10^{15} B	...	Facebook-Server
EB (Exabyte)	10^{18} B	...	Alle CERN-Daten
ZB (Zettabyte)	10^{21} B	...	Alle Daten ($\sim$ 100 ZB)
YB (Yottabyte)	10^{24} B	...	$\sim$ 2030?

Entwicklung der globalen Datenmenge

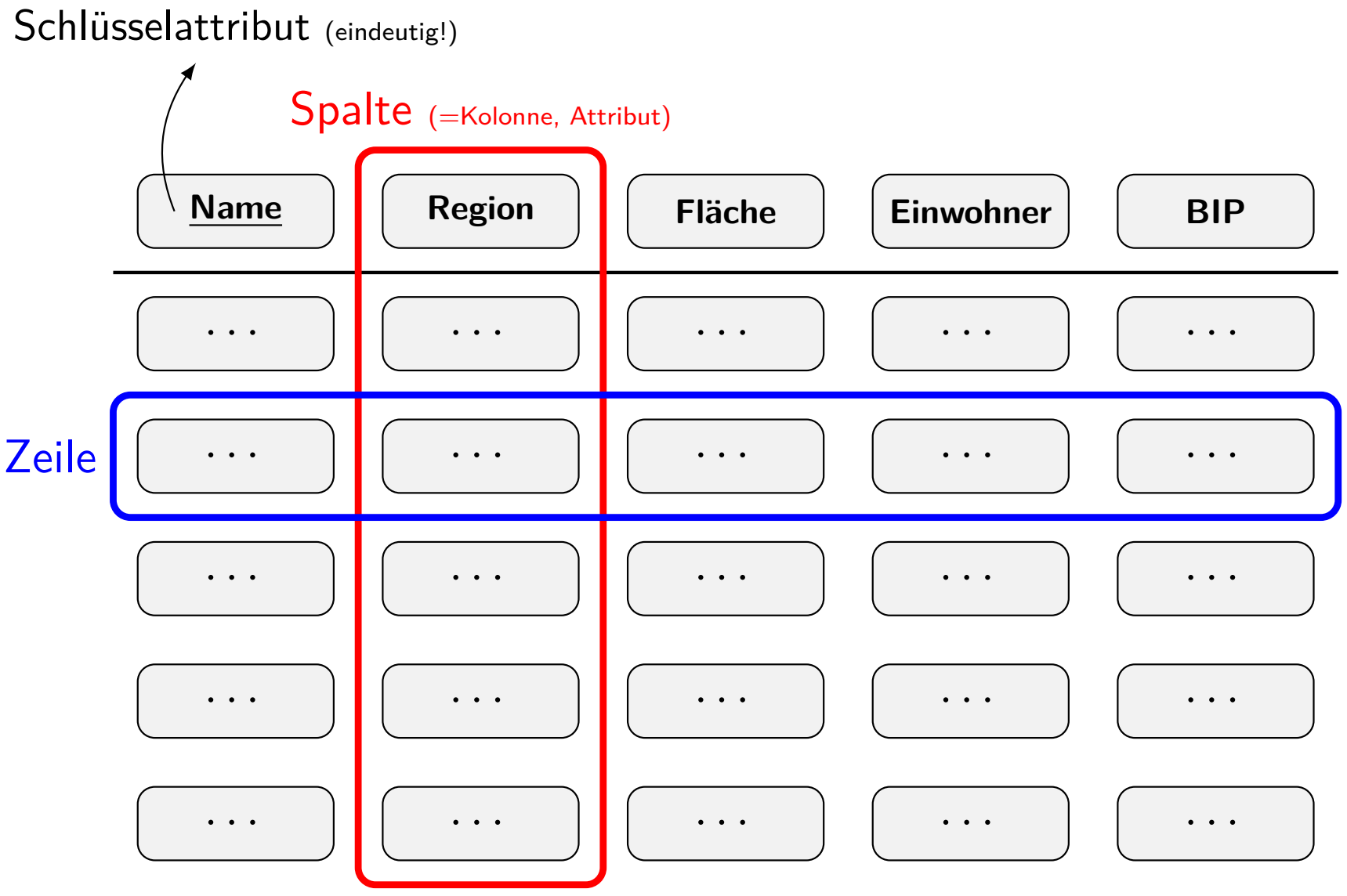


Beispiel-Datenbank: *Central Intelligence Agency* (CIA)

Name	Region	Fläche	Einwohner	BIP
Afghanistan	Asien	652	25.8M	21.0B
Albanien	Europa	28.7	3.49M	5.6B
Algerien	Afrika	2,381.7	31.2M	147.6B
Amerikanische...	Ozeanien	0.199	65.4K	0.15B
Andorra	Europa	0.468	66.8K	1.2B
Angola	Afrika	1,246.7	10.1M	11.6B
Anguilla	Mittelamerika	0.091	11.8K	0.088B
Antarktik	Antarktis	14M	0	0
Antigua und...	Mittelamerika	0.442	66.4K	0.524B
Argentinien	Südamerika	2,766.9	36.9M	367B
...	...	...	...	...

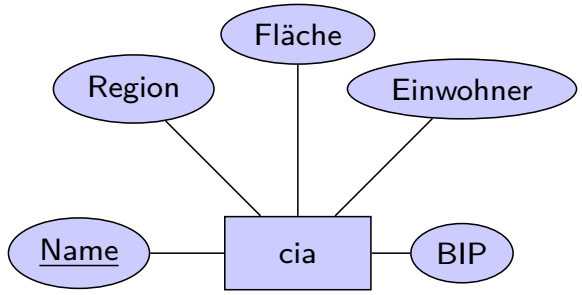


Typische Tabellenstruktur



SQL-Befehle

`SELECT [spalten] FROM [tabellenname]`

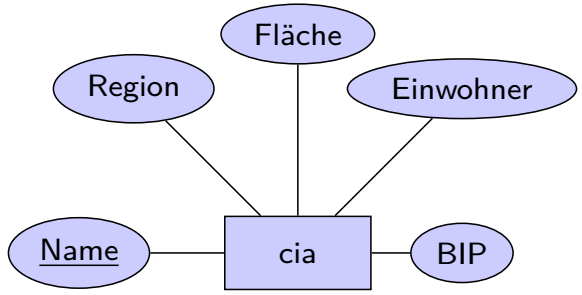



```
SELECT *
FROM cia;
```

Name	Region	Fläche	Einwohner	BIP
Afghanistan	Asien	652	25.8M	21.0B
Albanien	Europa	28.7	3.49M	5.6B
Algerien	Afrika	2,381.7	31.2M	147.6B
Amerikanische...	Ozeanien	0.199	65.4K	0.15B
Andorra	Europa	0.468	66.8K	1.2B
Angola	Afrika	1,246.7	10.1M	11.6B
Anguilla	Mittelamerika	0.091	11.8K	0.088B
Antarktik	Antarktis	14M	0	0
Antigua und...	Mittelamerika	0.442	66.4K	0.524B
Argentinien	Südamerika	2,766.9	36.9M	367B
...	...	...	...	...

SQL-Befehle

```
SELECT [spalten] FROM [tabellenname] WHERE [conditions]
```




```
SELECT Name, Region
FROM cia
WHERE Region = "Afrika";
```

Name	Region
Afghanistan	Asien
Albanien	Europa
Algerien	Afrika
Amerikanische...	Ozeanien
Andorra	Europa
Angola	Afrika
Anguilla	Mittelamerika
Antarktik	Antarktis
Antigua und...	Mittelamerika
Argentinien	Südamerika
...	...

→

Name	Region
Algerien	Afrika
Angola	Afrika
Benin	Afrika
Botswana	Afrika
Burkina Faso	Afrika
...	...
Sudan	Afrika
Swaziland	Afrika
Tanzania	Afrika
Togo	Afrika
Tunesien	Afrika

Lehrmittel: Cheat Sheet

Cheatsheet

Folgendes Cheatsheet ist basiert auf der Datenbank der Social-Media-Plattform InstaHub. In den ersten Befehlen wird insbesondere die Tabelle `users` verwendet. Die Tabelle enthält unter anderem folgende Informationen:

id	username	name	birthday	city	centimeters
1	niclas258	Niclas Schweizer	2001-01-31	Wremen	182
2	rafael54	Rafael Probst	2004-08-06	Leipzig	187
3	luis52	Luis Krüger	2004-12-15	Lautertal	173
...	...	...	...	...	...

Daneben enthält die Tabelle noch viele weitere Spalten mit Informationen zum Land, Geschlecht usw.

Spalten auswählen: SELECT

Auswahl aller Spalten der Tabelle:

```
SELECT *
FROM users
```

Auswahl der Spalten `city` und `gender` der Tabelle `users`:

```
SELECT city, gender
FROM users
```

Duplikate löschen: DISTINCT

Mit dem Zusatz `DISTINCT` werden Duplikate aus den Resultaten gelöscht.

```
SELECT DISTINCT gender
FROM users
```

Sortieren: ORDER BY

Mit dem Befehl `ORDER BY` können Resultate sortiert werden, nach einer oder mehreren Spalten. Mit folgendem Befehl erhalten wir die Namen aller Users, sortiert nach Stadt und nach Region:

```
SELECT name, city
FROM users
ORDER BY city ASC, name ASC
```

- `DESC` = absteigend (*descending*)
- `ASC` = aufsteigend (*ascending*)

Erste / Letzte Zeilen: LIMIT

Mit dem Befehl `LIMIT` können eine Tabelle auf die ersten n Zeilen beschränkt werden. Z.B. können die drei ersten Städte im Alphabet wie folgt abgefragt werden:

```
SELECT city
FROM users
ORDER BY city ASC
LIMIT 3
```

Zeilen Filtern: WHERE

Mit dem Befehl `WHERE` können die Resultate einer Abfrage nach eigenen Kriterien gefiltert werden:

```
SELECT name, city, gender
FROM users
WHERE gender="male"
```

Operatoren	Bedeutung
`=`	gleich (=)
`<>`	ungleich (≠)
`<`	kleiner als (<)
`<=`	kleiner oder gleich (≤)
`>`	grösser als (>)
`>=`	grösser oder gleich (≥)
`BETWEEN x AND y`	ein Wert zwischen x und y
`IN (wert1, wert2, ...)`	einer von mehreren Werten
`IS NULL`	Wert ist leer

Filter kombinieren: AND, OR

Mehrere Filter können mit `AND` („und“ → beides muss wahr sein) oder `OR` („oder“ → eine der beiden Bedingungen muss wahr sein) verbunden werden.

Users aus Leipzig, Berlin oder Hamburg, welche grösser als 170 sind:

```
SELECT name, city
FROM users
WHERE city IN ("Leipzig","Berlin",
    "Hamburg")
 AND centimeters > 170
```

Ungefähre Treffer: LIKE

Mit folgendem Befehl erhalten wir alle Städte, die mit “Be...” beginnen:

```
SELECT DISTINCT city
FROM users
WHERE city LIKE "Be%"
```

Das Prozentzeichen (%) ist ein sogenannter *Platzhalter* und steht für „irgendetwas kommt hier (oder nicht)“. In diesem Beispiel bedeutet das, dass 0, 1, oder mehr Zeichen auf das „Be...“ folgen können.

Aggregatsfunktionen

Mit Aggregatsfunktionen können Daten zusammengefasst werden, beispielsweise um zu zählen, wie viele Zeilen in einer Tabelle sind (`COUNT`), das Maximum / Minimum zu berechnen (`MAX` / `MIN`), um die Summe oder den Mittelwert einer Spalte zu berechnen (`SUM` / `AVG`) oder um die Länge eines Texts zu berechnen (`LENGTH`). Z.B. berechnet folgender Ausdruck die Anzahl Users in Leipzig:

```
SELECT city, COUNT(*)
FROM users
WHERE city = "Leipzig"
```

Rechnen und umbenennen: AS

Mit Spalten sowie Aggregatsfunktionen kann man rechnen, beispielsweise um ein Resultat durch eine Million zu dividieren. Zudem können Spalten-Titel mit dem Befehl `AS` umbenannt werden:

```
SELECT COUNT(*)/1e6 AS "Millionen
    Users"
FROM users
```

Gruppieren: GROUP BY

Aggregatsfunktionen können auch pro Gruppe verwendet werden, z.B. um die Anzahl Mitglieder in jeder Stadt zu berechnen:

```
SELECT city, COUNT(*) AS "Users
    pro Stadt"
FROM users
GROUP BY city
```

Filtern *nach* Gruppieren: HAVING

Mit `HAVING` können Resultate *nach dem* Gruppieren gefiltert werden. `WHERE` filtert *vor* dem Gruppieren und steht demnach immer vor einem `GROUP BY`.

Durchschnittliche Körpergrösse aller männlichen Mitglieder in jeder Stadt berechnen, danach auf Städte beschränken, in denen die Menschen durchschnittlich zwischen 150 und 155 gross sind:

```
SELECT city, AVG(centimeters) AS "
    Körpergrösse"
FROM users
WHERE gender = "male"
GROUP BY city
HAVING `Körpergrösse` BETWEEN 150
    AND 155
```

Texte werden immer innerhalb von Anführungszeichen (`""`) geschrieben. Spaltennamen, welche Spezialzeichen oder Abstände enthalten, werden innerhalb von *backticks* (``` `` ```) geschrieben.

- Viel Eigenständigkeit erwartet

Lehrmittel: Cheat Sheet

Cheatsheet

Folgendes Cheatsheet ist basiert auf der Datenbank der Social-Media-Plattform InstaHub. In den ersten Befehlen wird insbesondere die Tabelle `users` verwendet. Die Tabelle enthält unter anderem folgende Informationen:

id	username	name	birthday	city	centimeters
1	niclas258	Niclas Schweizer	2001-01-31	Wremen	182
2	rafael54	Rafael Probst	2004-08-06	Leipzig	187
3	luis52	Luis Krüger	2004-12-15	Lautertal	173
...	...	...	...	...	...

Daneben enthält die Tabelle noch viele weitere Spalten mit Informationen zum Land, Geschlecht usw.

Spalten auswählen: `SELECT`

Auswahl aller Spalten der Tabelle:

```
SELECT *
FROM users
```

Auswahl der Spalten `city` und `gender` der Tabelle `users`:

```
SELECT city, gender
FROM users
```

Duplikate löschen: `DISTINCT`

Mit dem Zusatz `DISTINCT` werden Duplikate aus den Resultaten gelöscht.

```
SELECT DISTINCT gender
FROM users
```

Sortieren: `ORDER BY`

Mit dem Befehl `ORDER BY` können Resultate sortiert werden, nach einer oder mehreren Spalten. Mit folgendem Befehl erhalten wir die Namen aller Users, sortiert nach Stadt und nach Region:

```
SELECT name, city
FROM users
ORDER BY city ASC, name ASC
```

- `DESC` = absteigend (*descending*)
- `ASC` = aufsteigend (*ascending*)

Erste / Letzte Zeilen: `LIMIT`

Mit dem Befehl `LIMIT` können eine Tabelle auf die ersten n Zeilen beschränkt werden. Z.B. können die drei ersten Städte im Alphabet wie folgt abgefragt werden:

```
SELECT city
FROM users
ORDER BY city ASC
LIMIT 3
```

Zeilen Filtern: `WHERE`

Mit dem Befehl `WHERE` können die Resultate einer Abfrage nach eigenen Kriterien gefiltert werden:

```
SELECT name, city, gender
FROM users
WHERE gender="male"
```

Operatoren	Bedeutung
`=`	gleich (=)
`<>`	ungleich ($\neq$)
`<`	kleiner als ($<$)
`<=`	kleiner oder gleich ($\leq$)
`>`	grösser als ($>$)
`>=`	grösser oder gleich ($\geq$)
`BETWEEN x AND y`	ein Wert zwischen x und y
`IN (wert1, wert2, ...)`	einer von mehreren Werten
`IS NULL`	Wert ist leer

Filter kombinieren: `AND`, `OR`

Mehrere Filter können mit `AND` („und“ → beides muss wahr sein) oder `OR` („oder“ → eine der beiden Bedingungen muss wahr sein) verbunden werden.

Users aus Leipzig, Berlin oder Hamburg, welche grösser als 170 sind:

```
SELECT name, city
FROM users
WHERE city IN ("Leipzig","Berlin",
    "Hamburg")
 AND centimeters > 170
```

Ungefähre Treffer: `LIKE`

Mit folgendem Befehl erhalten wir alle Städte, die mit “Be...” beginnen:

```
SELECT DISTINCT city
FROM users
WHERE city LIKE "Be%"
```

Das Prozentzeichen (%) ist ein sogenannter *Platzhalter* und steht für „irgendetwas kommt hier (oder nicht)“. In diesem Beispiel bedeutet das, dass 0, 1, oder mehr Zeichen auf das „Be...“ folgen können.

Aggregatsfunktionen

Mit Aggregatsfunktionen können Daten zusammengefasst werden, beispielsweise um zu zählen, wie viele Zeilen in einer Tabelle sind (`COUNT`), das Maximum / Minimum zu berechnen (`MAX` / `MIN`), um die Summe oder den Mittelwert einer Spalte zu berechnen (`SUM` / `AVG`) oder um die Länge eines Texts zu berechnen (`LENGTH`). Z.B. berechnet folgender Ausdruck die Anzahl Users in Leipzig:

```
SELECT city, COUNT(*)
FROM users
WHERE city = "Leipzig"
```

Rechnen und umbenennen: `AS`

Mit Spalten sowie Aggregatsfunktionen kann man rechnen, beispielsweise um ein Resultat durch eine Million zu dividieren. Zudem können Spalten-Titel mit dem Befehl `AS` umbenannt werden:

```
SELECT COUNT(*)/1e6 AS "Millionen
    Users"
FROM users
```

Gruppieren: `GROUP BY`

Aggregatsfunktionen können auch pro Gruppe verwendet werden, z.B. um die Anzahl Mitglieder in jeder Stadt zu berechnen:

```
SELECT city, COUNT(*) AS "Users
    pro Stadt"
FROM users
GROUP BY city
```

Filtern *nach* Gruppieren: `HAVING`

Mit `HAVING` können Resultate *nach dem* Gruppieren gefiltert werden. `WHERE` filtert *vor* dem Gruppieren und steht demnach immer vor einem `GROUP BY`.

Durchschnittliche Körpergrösse aller männlichen Mitglieder in jeder Stadt berechnen, danach auf Städte beschränken, in denen die Menschen durchschnittlich zwischen 150 und 155 gross sind:

```
SELECT city, AVG(centimeters) AS "
    Körpergrösse"
FROM users
WHERE gender = "male"
GROUP BY city
HAVING `Körpergrösse` BETWEEN 150
    AND 155
```

Texte werden immer innerhalb von Anführungszeichen (`""`) geschrieben. Spaltennamen, welche Spezialzeichen oder Abstände enthalten, werden innerhalb von *backticks* (`` ` ``) geschrieben.

- Viel Eigenständigkeit erwartet
- Automatische Lösungsvorschläge auf Moodle

Lehrmittel: Cheat Sheet

Cheatsheet

Folgendes Cheatsheet ist basiert auf der Datenbank der Social-Media-Plattform InstaHub. In den ersten Befehlen wird insbesondere die Tabelle `users` verwendet. Die Tabelle enthält unter anderem folgende Informationen:

id	username	name	birthday	city	centimeters
1	niclas258	Niclas Schweizer	2001-01-31	Wremen	182
2	rafael54	Rafael Probst	2004-08-06	Leipzig	187
3	luis52	Luis Krüger	2004-12-15	Lautertal	173
...	...	...	...	...	...

Daneben enthält die Tabelle noch viele weitere Spalten mit Informationen zum Land, Geschlecht usw.

Spalten auswählen: `SELECT`

Auswahl aller Spalten der Tabelle:

```
SELECT *
FROM users
```

Auswahl der Spalten `city` und `gender` der Tabelle `users`:

```
SELECT city, gender
FROM users
```

Duplikate löschen: `DISTINCT`

Mit dem Zusatz `DISTINCT` werden Duplikate aus den Resultaten gelöscht.

```
SELECT DISTINCT gender
FROM users
```

Sortieren: `ORDER BY`

Mit dem Befehl `ORDER BY` können Resultate sortiert werden, nach einer oder mehreren Spalten. Mit folgendem Befehl erhalten wir die Namen aller Users, sortiert nach Stadt und nach Region:

```
SELECT name, city
FROM users
ORDER BY city ASC, name ASC
```

- `DESC` = absteigend (*descending*)
- `ASC` = aufsteigend (*ascending*)

Erste / Letzte Zeilen: `LIMIT`

Mit dem Befehl `LIMIT` können eine Tabelle auf die ersten n Zeilen beschränkt werden. Z.B. können die drei ersten Städte im Alphabet wie folgt abgefragt werden:

```
SELECT city
FROM users
ORDER BY city ASC
LIMIT 3
```

Zeilen Filtern: `WHERE`

Mit dem Befehl `WHERE` können die Resultate einer Abfrage nach eigenen Kriterien gefiltert werden:

```
SELECT name, city, gender
FROM users
WHERE gender="male"
```

Operatoren	Bedeutung
`=`	gleich (=)
`<>`	ungleich ($\neq$)
`<`	kleiner als (<)
`<=`	kleiner oder gleich ($\leq$)
`>`	grösser als (>)
`>=`	grösser oder gleich ($\geq$)
`BETWEEN x AND y`	ein Wert zwischen x und y
`IN (wert1, wert2, ...)`	einer von mehreren Werten
`IS NULL`	Wert ist leer

Filter kombinieren: `AND`, `OR`

Mehrere Filter können mit `AND` („und“ → beides muss wahr sein) oder `OR` („oder“ → eine der beiden Bedingungen muss wahr sein) verbunden werden.

Users aus Leipzig, Berlin oder Hamburg, welche grösser als 170 sind:

```
SELECT name, city
FROM users
WHERE city IN ("Leipzig","Berlin",
    "Hamburg")
 AND centimeters > 170
```

Ungefähre Treffer: `LIKE`

Mit folgendem Befehl erhalten wir alle Städte, die mit “Be...” beginnen:

```
SELECT DISTINCT city
FROM users
WHERE city LIKE "Be%"
```

Das Prozentzeichen (%) ist ein sogenannter *Platzhalter* und steht für „irgendetwas kommt hier (oder nicht)“. In diesem Beispiel bedeutet das, dass 0, 1, oder mehr Zeichen auf das „Be...“ folgen können.

Aggregatsfunktionen

Mit Aggregatsfunktionen können Daten zusammengefasst werden, beispielsweise um zu zählen, wie viele Zeilen in einer Tabelle sind (`COUNT`), das Maximum / Minimum zu berechnen (`MAX` / `MIN`), um die Summe oder den Mittelwert einer Spalte zu berechnen (`SUM` / `AVG`) oder um die Länge eines Texts zu berechnen (`LENGTH`). Z.B. berechnet folgender Ausdruck die Anzahl Users in Leipzig:

```
SELECT city, COUNT(*)
FROM users
WHERE city = "Leipzig"
```

Rechnen und umbenennen: `AS`

Mit Spalten sowie Aggregatsfunktionen kann man rechnen, beispielsweise um ein Resultat durch eine Million zu dividieren. Zudem können Spalten-Titel mit dem Befehl `AS` umbenannt werden:

```
SELECT COUNT(*)/1e6 AS "Millionen
    Users"
FROM users
```

Gruppieren: `GROUP BY`

Aggregatsfunktionen können auch pro Gruppe verwendet werden, z.B. um die Anzahl Mitglieder in jeder Stadt zu berechnen:

```
SELECT city, COUNT(*) AS "Users
    pro Stadt"
FROM users
GROUP BY city
```

Filtern *nach* Gruppieren: `HAVING`

Mit `HAVING` können Resultate *nach dem* Gruppieren gefiltert werden. `WHERE` filtert *vor* dem Gruppieren und steht demnach immer vor einem `GROUP BY`.

Durchschnittliche Körpergrösse aller männlichen Mitglieder in jeder Stadt berechnen, danach auf Städte beschränken, in denen die Menschen durchschnittlich zwischen 150 und 155 gross sind:

```
SELECT city, AVG(centimeters) AS "
    Körpergrösse"
FROM users
WHERE gender = "male"
GROUP BY city
HAVING `Körpergrösse` BETWEEN 150
    AND 155
```

Texte werden immer innerhalb von Anführungszeichen (`""`) geschrieben. Spaltennamen, welche Spezialzeichen oder Abstände enthalten, werden innerhalb von *backticks* (``` `` ```) geschrieben.

- Viel Eigenständigkeit erwartet
- Automatische Lösungsvorschläge auf Moodle
- Nächstes Mal: *Murder Mystery*

Lehrmittel: Cheat Sheet

Cheatsheet

Folgendes Cheatsheet ist basiert auf der Datenbank der Social-Media-Plattform InstaHub. In den ersten Befehlen wird insbesondere die Tabelle `users` verwendet. Die Tabelle enthält unter anderem folgende Informationen:

id	username	name	birthday	city	centimeters
1	niclas258	Niclas Schweizer	2001-01-31	Wremen	182
2	rafael54	Rafael Probst	2004-08-06	Leipzig	187
3	luis52	Luis Krüger	2004-12-15	Lautertal	173
...	...	...	...	...	...

Daneben enthält die Tabelle noch viele weitere Spalten mit Informationen zum Land, Geschlecht usw.

Spalten auswählen: SELECT

Auswahl aller Spalten der Tabelle:

```
SELECT *
FROM users
```

Auswahl der Spalten `city` und `gender` der Tabelle `users`:

```
SELECT city, gender
FROM users
```

Duplikate löschen: DISTINCT

Mit dem Zusatz `DISTINCT` werden Duplikate aus den Resultaten gelöscht.

```
SELECT DISTINCT gender
FROM users
```

Sortieren: ORDER BY

Mit dem Befehl `ORDER BY` können Resultate sortiert werden, nach einer oder mehreren Spalten. Mit folgendem Befehl erhalten wir die Namen aller Users, sortiert nach Stadt und nach Region:

```
SELECT name, city
FROM users
ORDER BY city ASC, name ASC
```

- `DESC` = absteigend (*descending*)
- `ASC` = aufsteigend (*ascending*)

Erste / Letzte Zeilen: LIMIT

Mit dem Befehl `LIMIT` können eine Tabelle auf die ersten n Zeilen beschränkt werden. Z.B. können die drei ersten Städte im Alphabet wie folgt abgefragt werden:

```
SELECT city
FROM users
ORDER BY city ASC
LIMIT 3
```

Zeilen Filtern: WHERE

Mit dem Befehl `WHERE` können die Resultate einer Abfrage nach eigenen Kriterien gefiltert werden:

```
SELECT name, city, gender
FROM users
WHERE gender="male"
```

Operatoren	Bedeutung
=	gleich (=)
<>	ungleich (≠)
<	kleiner als (<)
<=	kleiner oder gleich (≤)
>	grösser als (>)
>=	grösser oder gleich (≥)
`BETWEEN x AND y`	ein Wert zwischen x und y
`IN (wert1, wert2, ...)`	einer von mehreren Werten
`IS NULL`	Wert ist leer

Filter kombinieren: AND, OR

Mehrere Filter können mit `AND` („und“ → beides muss wahr sein) oder `OR` („oder“ → eine der beiden Bedingungen muss wahr sein) verbunden werden.

Users aus Leipzig, Berlin oder Hamburg, welche grösser als 170 sind:

```
SELECT name, city
FROM users
WHERE city IN ("Leipzig","Berlin",
    "Hamburg")
 AND centimeters > 170
```

Ungefähre Treffer: LIKE

Mit folgendem Befehl erhalten wir alle Städte, die mit “Be...” beginnen:

```
SELECT DISTINCT city
FROM users
WHERE city LIKE "Be%"
```

Das Prozentzeichen (%) ist ein sogenannter *Platzhalter* und steht für „irgendetwas kommt hier (oder nicht)“. In diesem Beispiel bedeutet das, dass 0, 1, oder mehr Zeichen auf das „Be...“ folgen können.

Aggregatsfunktionen

Mit Aggregatsfunktionen können Daten zusammengefasst werden, beispielsweise um zu zählen, wie viele Zeilen in einer Tabelle sind (`COUNT`), das Maximum / Minimum zu berechnen (`MAX` / `MIN`), um die Summe oder den Mittelwert einer Spalte zu berechnen (`SUM` / `AVG`) oder um die Länge eines Texts zu berechnen (`LENGTH`). Z.B. berechnet folgender Ausdruck die Anzahl Users in Leipzig:

```
SELECT city, COUNT(*)
FROM users
WHERE city = "Leipzig"
```

Rechnen und umbenennen: AS

Mit Spalten sowie Aggregatsfunktionen kann man rechnen, beispielsweise um ein Resultat durch eine Million zu dividieren. Zudem können Spalten-Titel mit dem Befehl `AS` umbenannt werden:

```
SELECT COUNT(*)/1e6 AS "Millionen
    Users"
FROM users
```

Gruppieren: GROUP BY

Aggregatsfunktionen können auch pro Gruppe verwendet werden, z.B. um die Anzahl Mitglieder in jeder Stadt zu berechnen:

```
SELECT city, COUNT(*) AS "Users
    pro Stadt"
FROM users
GROUP BY city
```

Filtern *nach* Gruppieren: HAVING

Mit `HAVING` können Resultate *nach dem* Gruppieren gefiltert werden. `WHERE` filtert *vor* dem Gruppieren und steht demnach immer vor einem `GROUP BY`.

Durchschnittliche Körpergrösse aller männlichen Mitglieder in jeder Stadt berechnen, danach auf Städte beschränken, in denen die Menschen durchschnittlich zwischen 150 und 155 gross sind:

```
SELECT city, AVG(centimeters) AS "
    Körpergrösse"
FROM users
WHERE gender = "male"
GROUP BY city
HAVING `Körpergrösse` BETWEEN 150
    AND 155
```

Texte werden immer innerhalb von Anführungszeichen (`""`) geschrieben. Spaltennamen, welche Spezialzeichen oder Abstände enthalten, werden innerhalb von *backticks* (``) geschrieben.

- Viel Eigenständigkeit erwartet
- Automatische Lösungsvorschläge auf Moodle
- Nächstes Mal: *Murder Mystery*
- Feedback ist willkommen!

Lehrmittel: Skript

Als Ergänzung und Vertiefung zum Cheatsheet, sowie Challenge-Aufgaben

- Arbeit hauptsächlich mit Cheat-Sheets
- Feedback ist willkommen!

Auftrag: Eigene Tabellen erstellen & Abfragen schreiben

Aufgaben und **Cheatsheet** auf Moodle