OpenSchoolMaps: OpenStreetMap als Fallbeispiel einer Datenanalyse

OpenSchoolMaps.ch — Freie Lernmaterialien zu freien Geodaten und Karten

Stefan Keller, Rapperswil, Juni 2026

Dies ist ein Workshop-Tutorial (ein digitales Lernobjekt) für Interessierte im Fach Informatik (Data Engineering inklusive Sekundarstufe II/Maturand:innen) mit eigenständigen Lerneinheiten, das dafür konzipiert ist, den selbstgesteuerten Lernprozess zu unterstützen. Dabei geht es um das Modellieren, Abfragen und Prüfen von Geodaten in einem Quartier am Beispiel von Street Art, Restaurants und Trottoirs aus OpenStreetMap. Die Bearbeitungsdauer beträgt ca. 3 Stunden.

Graffiti als Street Art mit Strassen-Restaurant. (Quelle: Generated by AI / Gemini)

Einleitung

OpenStreetMap (OSM) ist das "Wikipedia der Geodaten" – eine frei verfügbare, kollaborativ erstellte Weltkarte. Mit über sieben Millionen registrierten Nutzer:innen bildet sie das Fundament für fast alle gängigen Navigationsanwendungen wie Garmin, Apple Maps oder Komoot. Allein in der Schweiz tragen täglich über 100 Personen Informationen zu Strassen, Gebäuden, Restaurants und Tausenden weiteren Objekten bei.

Doch was auf der Karte wie ein nahtloses System erscheint, ist technisch gesehen eine komplexe Herausforderung der Datenanalyse. Angenommen, wir wollen ein eigenes Projekt starten und Street Art im Quartier kartieren: Wie speichern wir solch vielfältige reale Objekte in einer Datenbank? Wie stellen wir präzise Abfragen, und woran erkennen wir, ob die Daten vollständig sind?

Hier setzt unser Workshop an. Die Leitfrage lautet: Wie gut sind Street Art, Restaurants und Trottoirs im Quartier in OSM tatsächlich erfasst?

Um diese Frage zu beantworten, tauchen wir in das Thema Data Engineering ein. Wir ergründen, wie

reale Gegebenheiten in OSM modelliert werden, wie daraus eine nutzbare Datenstruktur entsteht und wie wir diese mittels SQL abfragen und kritisch auf Qualität prüfen.

Nach dem Workshop sind die Teilnehmenden in der Lage, Geodaten aus OpenStreetMap souverän zu modellieren, mit SQL zu extrahieren und ihre Qualität zu bewerten.

OpenStreetMap als passendes Fallbeispiel

OSM ist kein gewöhnliches Datenbanksystem. Es verwendet ein flexibles Modell, bei dem die Bedeutung eines Objekts nicht durch feste Tabellenspalten definiert wird, sondern durch freie Schlüssel-Wert-Paare. Schlüssel-Wert-Paare werden Tags genannt. Das macht OSM zu einem interessanten Fallbeispiel für die Datenanalyse, weil es grundlegende Fragen der Datenverwaltung aufwirft:

- Wie bildet man ein flexibles, semi-strukturiertes Schema in ein strukturiertes Schema einer Datenbank ab?
- Wie schreibt man SQL-Abfragen auf einem solchen Datenmodell?
- Was ist Datenqualität, und wie lässt sie sich automatisiert prüfen?

Voraussetzungen

- Grundlagen Datenbanken: Konzept von Tabellen, Spalten, Primärschlüssel
- Grundlagen SQL: `SELECT`, `FROM`, `WHERE`, einfache Filterung
- Grundlagen OSM: Verständnis, was OpenStreetMap ist (kurze Demo oder dieses Tutorial genügt)
- Grundlagen Geovisualisierung: Kartenerstellung z.B. mit DataWrapper, uMap oder QGIS

- Um SQL besser kennenzulernen, empfehlen wir das preisgekrönte Educational Game "SQL Island" wie es im OpenSchoolMaps-Tutorial Ein sanfter Einstieg in Spatial SQL mit PostGIS beschrieben wird.
- Und wenn Grundlagen der Geovisualisierung fehlen, empfehlen wir das OpenSchoolMaps-Tutorial Karten im Web veröffentlichen.

Der rote Faden

Dieser Kurs hat folgenden Ablauf: Wir starten bei der realen Welt und arbeiten uns Schritt für Schritt bis zur automatisierten Qualitätsprüfung und Visualisierung vor.

```
Realwelt → OSM-Objekt → Tags & Geometrie → Strukturiertes Modell → SQL-Abfrage
→ Qualitätsprüfung → Visualisierung
```

Die Kernidee: Reale Objekte fallen nicht automatisch in saubere Tabellen. Erst durch bewusste Modellierungsentscheidungen entstehen aus flexiblen Tags und Geometrien abfragbare Datenbankstrukturen.

Für jeden Schritt gibt es eine Lerneinheit, die jeweils offene Aufgabenstellungen enthalten, teilweise gefolgt von einem Tipp.

Wichtige Konzepte

- Entität: Ein Street-Art-Objekt, ein Restaurant, ein Trottoirabschnitt
- Attribut: `name`, `amenity`, `tourism`, `cuisine`, `opening_hours`, `surface`
- Geometrie: Punkt (`Node`), Linie (`Way`), Fläche (geschlossener `Way`)
- Schema-on-read: Ansatz, bei dem die Bedeutung eines Objekts nicht durch feste Tabellenspalten, sondern durch die Interpretation der Tags beim Lesen bestimmt wird
- Normalisierung: Tags als eigene Key-Value-Tabelle statt vieler leerer Spalten
- Semi-strukturierte Daten: Tags als `jsonb`-Feld in PostgreSQL/PostGIS
- SQL-Abfrage: `WHERE tags->>'amenity' = 'restaurant'`
- Datenqualität: Fehlende Tags, Dubletten, uneinheitliche Modellierung

Overpass API

In diesem Tutorial verwenden wir die Overpass-Schnittstelle. Dabei unterscheiden wir folgende vier:

- *Overpass API:* Eine Schnittstelle (API), mit der man gezielt OpenStreetMap-Daten abfragen kann, ohne den riesigen OSM-Datensatz zuerst herunterladen zu müssen. (dev.overpass-api.de)
- *Overpass Turbo:* Eine Webseite, auf der man Overpass-Abfragen schreiben, testen und die Ergebnisse direkt auf einer Karte sehen kann. (overpass-turbo.eu)
- *Overpass QL:* Die Abfragesprache, mit der man der Overpass API mitteilt, welche OSM-Objekte man sucht, zum Beispiel Restaurants, Wege oder Gebäude.
- *Postpass:* Ein ähnlicher Dienst wie Overpass API, aber mit SQL als Abfragesprache. (postpass.geofabrik.de)

Für alle Abfragen in diesem Workshop verwenden wir overpass-turbo.eu. Overpass Turbo ist eine webbasierte Oberfläche zum Abfragen von OSM-Daten. Normalerweise werden dort Abfragen in der Overpass Query Language (Overpass QL) geschrieben. Die Daten werden regelmässig aus OSM importiert.

Wir nutzen hier statt Overpass QL den Postpass-Server, der es erlaubt, SQL zu schreiben.

Um Postpass besser kennenzulernen, gibt es neben der offiziellen Postpass-Homepage auch ein OpenSchoolMaps-Tutorial zu "OpenStreetMap-Daten extrahieren mit SQL und Postpass".

Das Untersuchungsgebiet: Seefeld (Zürich)

Als Beispiel-Bounding-Box verwenden wir das Quartier Seefeld in Zürich. In Spatial SQL wird diese Bounding Box wie folgt als Geometrie ausgedrückt:

```
ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
```

Die vier Argumente folgen der Standard-Syntax (xmin, ymin, xmax, ymax), was in diesem Fall der Reihenfolge (MinLon, MinLat, MaxLon, MaxLat) entspricht (lat/lon bezeichnet Breitengrad und Längengrad, wobei die Breitengrade "die horizontalen Linien" auf dem Globus sind).

Die Zahl `4326` ist der SRID-Code (Spatial Reference ID) für das Koordinatensystem WGS 84, das von GPS und OSM verwendet wird. Für andere Quartiere können die Koordinaten entsprechend angepasst werden.

Wenn ihr den Kartenausschnitt in overpass-turbo auf das gewünschte Gebiet eingestellt habt, kann `{{bbox}}` als Platzhalter für die aktuelle Kartenansicht verwendet werden. Mit bboxfinder.com findet man rasch eigene bbox-Parameter.

Lerneinheit 1 – Von der Realwelt zum OSM-Objekt

Dauer: ca. 30 Minuten

Lernziele

- Den Unterschied zwischen einem realen Objekt und seiner digitalen Repräsentation erklären können.
- Die OSM-Elemente Punkt (`Node`), Linie (`Way`) und Fläche (`Way` oder `Relation`) kennen.
- Für ein gegebenes Realweltobjekt den passenden OSM-Typ und typische Tags bestimmen.
- Erkennen, warum mehrere Modellierungsoptionen für dasselbe Objekt zu Datenqualitätsproblemen wie Dubletten führen können.

Entitäten in der realen Welt

Bevor wir über Datenbanken sprechen können, müssen wir klären, was wir eigentlich erfassen wollen. In einem Quartier gibt es unzählige Dinge: Wandmalereien, Restaurants, Strassen, Bäume, Briefkästen. In der Datenbanksprache nennt man solche eindeutig abgrenzbaren Dinge Entitäten.

Eine Entität ist ein eindeutig identifizierbares Objekt der realen Welt, das uns interessiert und über das wir Daten speichern wollen. Beispiel: ein Wandbild an der Hausfassade an der Seefeldstrasse in Zürich.

Ein Attribut ist eine Eigenschaft dieser Entität. Beispiel: Das Wandbild hat einen Titel, einen Standort und einen bekannten Künstler. Jede Entität ist einmalig und wird durch eine eindeutige ID unterschieden – auch wenn zwei Kunstwerke identische Attribute hätten.

Von der Entität zum OSM-Objekt

OSM kennt drei Grundtypen für die Geometrie eines Objekts: Punkte, Linien und Flächen. Die Wahl hängt davon ab, wie gross und wie ausgedehnt das Objekt in der Realität ist.

Realweltobjekt	OSM-Typ	Begründung
Street Art	Node (Punkt)	Kleines, punktförmiges Objekt ohne relevante Ausdehnung auf der Karte
Restaurant	Node (Punkt)	Wenn Gebäudeumrisse nicht relevant sind oder in Einkaufszentren oder Hotels mit mehreren Restaurants
Restaurant (mit Gebäude)	Way oder Relation (Fläche)	Flächen-Grundriss des Gebäudes
Trottoir (an Strassenachse angehängt)	Tag an einem Way	Strassenachse mit `sidewalk=left/right/both` an der Strasse selbst
Trottoir (Objekt mit eigener Geometrie)	Way (Linie)	Ein separat erfasster Gehweg neben der Strasse

Restaurants sind in OSM also entweder als Punkt oder als Fläche erfasst – das müssen wir bei der Abfrage beachten. Es kann aber auch dazu führen, dass Restaurants doppelt erfasst werden, ein typisches Datenqualitätsproblem, das wir in Lerneinheit 5 untersuchen.

Aufgaben

Öffnet openstreetmap.org und navigiert zum Seefeld oder zu einem euch bekannten Quartier.

1. Findet je ein Beispiel für ein Street-Art-Objekt (`tourism=artwork`), ein Restaurant (`amenity=restaurant`) und ein Trottoir (`footway=sidewalk` oder `sidewalk=*`).
2. Klickt auf das Objekt und notiert, welchen OSM-Typ es hat (`Node` / `Way`).
3. Welche Attribute (`Tags`) sind angegeben? Welche fehlen eurer Meinung nach?

Mit dem Werkzeug Abfragen (Fragezeichen-Icon) auf openstreetmap.org lässt sich direkt auf ein Objekt klicken und dessen vollständige Tag-Liste ansehen – ohne den iD-Editor öffnen zu müssen.

Lerneinheit 2 – OSM-Tags und Schema-on-read

Dauer: ca. 30 Minuten

Lernziele

- Das Prinzip von Schema-on-write und Schema-on-read erklären und anhand von OSM illustrieren
- Die Tag-Struktur `key=value` in OSM verstehen und gebräuchliche Tags benennen
- Die Konsequenzen flexibler Tagging-Systeme für Abfragen und Datenqualität einschätzen

Wie OSM Objekte beschreibt: Tags

In OSM wird jedes Objekt über Tags beschrieben – einfache Schlüssel-Wert-Paare der Form `key=value`. Ein Street-Art-Objekt könnte zum Beispiel – nebst der Lage – folgende Tags tragen:

```
tourism      = artwork
name         = Seefeld-Mural
artwork_type = mural
artist_name  = unbekannt
start_date   = 2021
```

Es gibt keine Pflicht, bestimmte Tags zu setzen – ausser dem Haupttag, der das Objekt grundsätzlich klassifiziert (hier: `tourism=artwork`). Alle weiteren Tags sind optional (Nebentags) und hängen davon ab, ob jemand die Zeit und das Wissen hatte, sie einzutragen.

Schema-on-write und Schema-on-read

In einer klassischen relationalen Datenbank legt man das Schema vor der Dateneingabe fest: Man definiert Tabellen und Spalten, und jede Zeile muss sich an diese Struktur halten. Das nennt man Schema-on-write.

OSM hingegen folgt dem Prinzip Schema-on-read: Es gibt kein fixes Spaltenschema. Die Bedeutung eines Objekts ergibt sich erst beim Lesen – durch die Interpretation der Tags. Das macht OSM extrem flexibel, aber auch weniger konsistent.

Merkmal	**Schema-on-write (klassisch)**	**Schema-on-read (OSM-Stil)**
Schema	Vor dem Einfügen definiert	Beim Lesen interpretiert
Flexibilität	Gering – feste Spalten	Hoch – beliebige Tags
Konsistenz	Garantiert durch die Datenbank	Abhängig von den Kartierenden
Abfragen	Direkt auf Spalten	Auf Schlüssel in einem Tags-Feld
Fehlende Werte	NULL in einer Spalte	Tag fehlt vollständig

Ein wichtiger Unterschied: Wenn ein Street-Art-Objekt kein Tag `artist_name` trägt, bedeutet das nicht, dass der Künstler unbekannt ist. Es bedeutet: Diese Information wurde noch nicht erfasst. Bei Datenqualitätsanalysen ist dieser Unterschied entscheidend.

Typische Tags unserer drei Objekte

Objekt	Geometrie	Typische Tags	Häufig fehlend
Street Art	Punkt	`tourism=artwork`, `name=`, `artwork_type=`, `artist_name=*`	`artist_name`, `artwork_type`, `start_date`
Restaurant	Punkt oder Fläche	`amenity=restaurant`, `name=`, `cuisine=`, `opening_hours=`, `website=`	`cuisine`, `opening_hours`, `phone`
Trottoir (Linie)	Linie	`highway=footway`, `footway=sidewalk`, `surface=`, `width=`	`surface`, `width`, `lit`
Trottoir (Attribut)	Attribut an Strasse	`sidewalk=both/left/right/no`	Detaileigenschaften fehlen ganz

Street Art sind Kunstwerke an Hausfassaden, "Kunst am Bau", und gehören zur zeitgenössischen Kunst (vgl. Zürich: https://www.zuerich.com/de/kunst-kultur/kunst-im-oeffentlichen-raum/street-art). Sie sind eine spezielle Form von Kunst im öffentlichen Raum.

Kunst im öffentlichen Raum sind Werke wie Skulpturen, Wandbilder oder Installationen, die sichtbar und frei zugänglich an Orten wie Plätzen, Strassen, Parks oder an Gebäuden stehen (vgl. z.B. Zürich: https://www.zuerich.com/de/kunst-kultur/kunst-im-oeffentlichen-raum). Falls es keine Street Art in eurem Quartier gibt, können auch andere Kunstwerke untersucht werden.

Aufgaben

Untersucht auf openstreetmap.org mehrere Restaurants im Seefeld oder in einem anderen Quartier:

1. Welche Tags tragen diese Restaurants? Erstellt eine Tabelle mit den gefundenen Keys.
2. Wie viele der gefundenen Restaurants haben einen `cuisine`-Tag? Wie viele haben `opening_hours`?
3. Diskutiert: Warum gibt es so grosse Unterschiede in der Tag-Vollständigkeit?

Das OSM-Wiki unter wiki.openstreetmap.org/wiki/Map_features listet alle gängigen Tags mit Beschreibungen und Beispielen – ideal als Referenz beim Vergleich verschiedener Objekte.

Lerneinheit 3 – Strukturiertes Modell ableiten

Dauer: ca. 40 Minuten

Lernziele

- Verstehen, warum OSM-Rohdaten nicht direkt für analytische SQL-Abfragen geeignet sind
- Drei Modellierungsansätze vergleichen: breite Tabelle, normalisiert und Hybridmodell mit JSONB
- Das Hybridmodell konzeptionell aufbauen und begründen
- Den Begriff Normalisierung an einem konkreten Beispiel erklären

Das Problem: OSM-Rohdaten sind nicht direkt in GIS analysierbar

OSM-Rohdaten liegen im XML- oder PBF-Format vor und bestehen aus Nodes, Ways und Relations mit beliebig vielen Tags. Dieses Format eignet sich schlecht für analytische Fragen wie "Welche Restaurants ohne `cuisine`-Tag gibt es im Seefeld?".

Wir brauchen deshalb einen Datenbank-Transfer: Die OSM-Daten werden in eine relationale Datenbank importiert, wo wir sie mit SQL abfragen können. Genau das macht Postpass: Es importiert laufend aktuelle OSM-Daten in eine Datenbank (PostgreSQL/PostGIS) und macht sie über SQL zugänglich. Tags werden dabei als `jsonb`-Feld gespeichert.

Drei Modellierungsansätze

Wie bilden wir die flexiblen OSM-Tags auf eine strukturierte Datenbank ab? In technischen Worten: Wie ist die Schema-Abbildung?

OSM speichert alles in drei Elementtypen (Node, Way, Relation) mit beliebigen Tags. Eine GIS-Tabelle erwartet aber feste Spalten und einen klaren Geometrietyp. Die Schema-Abbildung muss unter anderem drei Fragen beantworten:

1. Welche Tabelle? — Das Haupt-Tag (`amenity=restaurant`, `highway=residential`) bestimmt, wohin das Element gehört.
2. Welche Geometrie? — Aus der Elementart und dem Kontext wird abgeleitet: Punkt, Linie oder Fläche. Manchmal ist es nötig, Elemente, die als Fläche erfasst sind, Punkten zuzuordnen – insbesondere bei Points of Interest (POI).
3. Welche Spalten? — Aus den vielen Sekundärtags werden nur die relevanten als Spalten übernommen; der Rest wird weggeworfen oder in einer JSON-Spalte aufbewahrt.

Es gibt mindestens drei grundsätzliche Ansätze. Postpass hat sich für Ansatz 3 (Hybridmodell) entschieden. Wer diese Diskussion abkürzen möchte, kann direkt zu Ansatz 3 und dann zu Lerneinheit 4 springen.

Ansatz 1: Breite Tabelle

Jeder mögliche Key wird zur eigenen Spalte:

```
feature(osm_id, osm_type, geom, amenity, tourism, name, cuisine,
```

```
        opening_hours, website, phone, artwork_type, artist_name,
        surface, width, ...)
```

Vorteil: Die Schema-Abbildung ist automatisch. Nachteil: Die meisten Felder sind `NULL`. Ein Street-Art-Objekt braucht keine `cuisine`-Spalte; ein Restaurant hat keinen `artwork_type`-Wert. Mit Tausenden möglicher OSM-Tags wäre die Tabelle absurd breit und kaum wartbar.

Dieser Ansatz ist nicht zu empfehlen – kann aber manchmal nicht vermieden werden. Das Problem sieht man anschaulich, wenn man Werkzeuge nutzt, die GeoJSON visualisieren – wie z.B. geojson.io – und von der JSON-Darstellung zur Tabellen-Darstellung wechselt.

Ansatz 2: Normalisierte Tabellen nach Themen

Es werden Entitätsklassen gebildet, mit denen die Welt strukturiert werden soll. Jeder Tabelle werden OSM-Elemente mit ausgewählten Tags zugeordnet.

```
building_polygon (
  feature_id    VARCHAR(20) PRIMARY KEY, -- z.B. N1234567890
  osm_type      TEXT,   -- 'way', 'relation'
  osm_id        BIGINT,
  name          TEXT,   -- direkt aus Tags extrahiert
  category      TEXT,   -- z.B. Gebäude, Schule, Behörde
  geom          GEOMETRY(POLYGON)
)
street_line (
  feature_id    VARCHAR(20) PRIMARY KEY, -- z.B. N1234567890
  osm_type      TEXT,   -- 'way'
  osm_id        BIGINT,
  name          TEXT,   -- direkt aus Tags extrahiert
  category      TEXT,   -- z.B. Fussweg, Strasse, Autobahn
  geom          GEOMETRY(LINESTRING)
)
poi_point (
  feature_id    VARCHAR(20) PRIMARY KEY, -- z.B. N1234567890
  osm_type      TEXT,    -- 'node'
  osm_id        BIGINT,
  name          TEXT,   -- direkt aus Tags extrahiert
  opening_hours TEXT,   -- gemäss OH spec. im Wiki
  main_category TEXT,   -- z.B. food
  sub_category  TEXT,   -- aus Tag-Aggregationen zusammengestellt
  geom          GEOMETRY(POINT)
)
poi_polygon (
  feature_id    VARCHAR(20) PRIMARY KEY, -- z.B. N1234567890
  osm_type      TEXT,    -- 'way', 'relation'
  osm_id        BIGINT,
  name          TEXT,   -- direkt aus Tags extrahiert
  main_category TEXT,   -- z.B. food
  sub_category  TEXT,   -- aus Tag-Aggregationen zusammengestellt
```

```
  geom          GEOMETRY(POLYGON)
)
... etc.
```

Vorteil: Keine NULL-Werte, korrekt normalisiert. Nachteil: Schema-Abbildung mit Informationsverlust, da nicht alle Tags und Werte abgebildet werden. Aufwändige Schema-Abbildung und wiederkehrende Anpassungen: Wann lohnt sich eine eigenständige Tabelle? Was gehört in `main_category`, was in `sub_category` – angesichts der grossen Menge an POI-Klassen?

Dieser Ansatz wurde z.B. im Projekt "Layercake" bzw. "OpenStreetMap Data in Layered GIS Format" der Firma Geofabrik gewählt. Die Postpass-SQL-Abfragen lassen sich gut an diese Datenmodelle anpassen, wobei die Tabellennamen angepasst werden müssen und alle `->>`-Operatoren durch normale Attribute mit dem `=`-Zeichen ersetzt werden müssen.

Ansatz 3: Hybridmodell

Ein Kompromiss: Die wichtigsten Attribute werden als eigene Spalten extrahiert, alle übrigen Tags als `jsonb`-Feld gespeichert.

```
geometrycollection_feature (
  osm_type      TEXT,     -- 'node', 'way', 'relation'
  osm_id        BIGINT,
  geom          GEOMETRY, -- Koordinaten (PostGIS)
  name          TEXT,     -- direkt aus Tags extrahiert
  tags          JSONB     -- alle weiteren Tags als JSON-Objekt
)
```

Wir nennen die Tabelle `geometrycollection_feature`, da die Geometrie-Spalte alle drei Typen (Punkt, Linie, Fläche) aufnehmen kann.

Vorteil: Flexibel und vollständig – keine Tags gehen verloren, und häufig benötigte Felder wie `name` sind direkt abfragbar. Durch den JSONB-Typ lässt sich dennoch gezielt auf einzelne Tags zugreifen. Nachteil: SQL-Abfragen auf JSONB sind weniger intuitiv als auf normale Spalten, und die Datenqualität hängt weiterhin von den Kartierenden ab – die Datenbank erzwingt keine Konsistenz.

JSONB ist ein Datentyp in PostgreSQL für JSON-Daten in binärer, indizierbarer Form. Im Gegensatz zu einer reinen Textspalte kann PostgreSQL damit direkt auf einzelne JSON-Schlüssel zugreifen. Der Operator `->>'key'` liefert den Wert eines Schlüssels als Text. Der Operator `? 'key'` prüft, ob ein Schlüssel überhaupt vorhanden ist.

Bei Postpass wurde dieser Ansatz gewählt, mit einigen Optimierungen (z.B. werden Node, Way, Relation mit N, W, R abgekürzt) und dadurch, dass aus der einzigen Tabelle geometrycollection_feature drei Tabellen verwendet werden mit dem entsprechenden Geometrie-Typ (point, linestring, polygon) im Tabellennamen. Das vollständige Schema von Postpass findet sich im Repository.

Aufgaben

Ihr habt soeben drei verschiedene Arten kennengelernt, OSM-Daten in einer Datenbank zu speichern. Ziel dieser Aufgabe ist es, die Ansätze anhand eines konkreten Beispiels zu vergleichen und zu beurteilen.

Ausgangssituation: Im Quartier Seefeld gibt es ein Restaurant mit folgenden OSM-Tags:

```
amenity       = restaurant
name          = Kronenhalle
cuisine       = swiss
opening_hours = Mo-Sa 12:00-24:00
website       = https://kronenhalle.com
phone         = +41 44 251 66 69
```

1. Ansatz 1 – Breite Tabelle: Skizziert, wie dieser Datensatz in einer breiten Tabelle aussehen würde. Nehmt an, die Tabelle hat ausserdem die Spalten `tourism`, `artwork_type`, `artist_name`, `surface`, `width`. Welche Felder wären `NULL`? Was passiert, wenn ein neues Restaurant mit dem Tag `wheelchair=yes` hinzukommt?
2. Ansatz 2 – Normalisierte Tabellen: In welche Tabelle aus dem Beispiel oben (`building_polygon`, `street_line`, `poi_point` oder `poi_polygon`) würde das Restaurant einsortiert? Welche Tags gehen dabei verloren?
3. Ansatz 3 – Hybridmodell: Zeigt, wie der Eintrag in der Tabelle `geometrycollection_feature` aussehen würde. Schreibt dazu den `name`-Wert als direkte Spalte und die restlichen Tags als JSONB-Objekt. Wie würde eine SQL-Abfrage lauten, die alle Restaurants mit Küchenangabe (`cuisine`) sucht?
4. Diskussion: Welcher Ansatz eignet sich am besten für den Workshop, und warum? Nennt je einen Vor- und einen Nachteil des Hybridmodells aus eurer eigenen Perspektive.

Das vollständige Postpass-Schema ist unter github.com/woodpeck/postpass-ops einsehbar – dort zeigt sich genau, welche Spalten und Views verfügbar sind, bevor wir eigene Abfragen entwerfen.

Lerneinheit 4 – SQL-Abfragen formulieren

Dauer: ca. 35 Minuten

Lernziele

- SQL-Abfragen auf Postpass-Views schreiben, die JSONB-Tags und Geometriefilter verwenden
- Den JSONB-Operator `->>'key'` und den Existenz-Operator `? 'key'` anwenden
- Abfragen auf overpass-turbo.eu mit der Postpass-Kopfzeile ausführen

Zur Erinnerung: Dieses Tutorial setzt voraus, dass man SQL-Grundlagen kennt. Für

> diejenigen, die SQL kennenlernen möchten, gibt es einen Hinweis im Kapitel Voraussetzungen oben.

Aufbau einer Postpass-SQL-Abfrage auf overpass-turbo.eu

Postpass stellt unter anderem folgende Tabellen und Views (vorbereitete, kombinierte Tabellen) bereit:

- `postpass_point` – Alle Objekte vom Typ Node (Punkte)
- `postpass_line` – Alle Objekte vom Typ Way, die Linien sind
- `postpass_polygon` – Alle Objekte vom Typ Way oder Relation, die Flächen sind
- `postpass_pointpolygon` – View der Vereinigung von Punkten und Flächen, nützlich z.B. für POIs wie Restaurants

Das vollständige Postpass-Schema findet sich unter diesem Link.

Postpass-Abfragen werden durch eine Kopfzeile eingeleitet, die den SQL-Server festlegt. Dann folgt die eigentliche SQL-Abfrage – ohne Strichpunkt am Schluss, wie es oft üblich ist.

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}
SELECT ...
FROM ...
WHERE ...
```

Die Kopfzeile `{{data:sql,server=⋯}}` muss immer auf der ersten Zeile stehen. Sie ist eine Anweisung an overpass-turbo, die nachfolgende SQL-Abfrage an den Postpass-Server weiterzuleiten.

Abfrage 1: Street Art im Seefeld finden

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}
SELECT
  osm_id,
  tags->>'name'         AS name,
  tags->>'artwork_type' AS artwork_type,
  tags->>'artist_name'  AS artist_name,
  geom
FROM postpass_point
WHERE tags->>'tourism' = 'artwork'
  AND geom && ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
```

Der Operator `&&` in PostGIS prüft, ob sich zwei Geometrien in ihren Bounding Boxes überlappen. Er ist sehr schnell, weil PostGIS dafür einen räumlichen Index verwendet. `ST_MakeEnvelope(west, süd, ost, nord, srid)` erstellt ein Rechteck aus den vier Eckkoordinaten.

Wenn der Tag `name` bei einem Objekt fehlt, liefert `tags->>'name'` den Wert `NULL` zurück – kein Fehler, sondern fehlende Information.

Abfrage 2: Restaurants mit Name und Küche

Restaurants kommen in OSM sowohl als Punkte als auch als Flächen vor. Deshalb verwenden wir den View `postpass_pointpolygon`, der beide vereint.

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}
SELECT
  osm_id,
  osm_type,                    -- 'node' oder 'way' – zeigt Punkt vs. Fläche
  tags->>'name'     AS name,
  tags->>'cuisine'  AS cuisine,
  geom
FROM postpass_pointpolygon
WHERE tags->>'amenity' = 'restaurant'
  AND geom && ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
```

Die Spalte `osm_type` zeigt, ob ein Restaurant als Node (`'node'`) oder als Flächenway (`'way'`) erfasst wurde. Kommen beide für dasselbe Gebäude vor, liegt eine Dublette vor.

Abfrage 3: Gehwege und Trottoirs

Trottoirs können auf zwei Arten in OSM modelliert sein: als eigene Linie mit `footway=sidewalk` oder als Attribut `sidewalk=*` an der Strasse selbst. Beide Varianten sollte die Abfrage erfassen.

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}
SELECT
  osm_id,
  tags->>'footway'   AS footway_typ,
  tags->>'sidewalk'  AS sidewalk_attribut,
  tags->>'surface'   AS belag,
  geom
FROM postpass_line
WHERE geom && ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
  AND (
    tags->>'footway' = 'sidewalk'     -- eigene Trottoir-Linie
    OR tags ? 'sidewalk'              -- Strasse mit sidewalk-Attribut
    OR tags->>'highway' = 'footway'   -- allgemeiner Fussweg
  )
```

Der Operator `? 'key'` gibt `TRUE` zurück, wenn der JSON-Schlüssel im Tags-Objekt vorhanden ist – unabhängig davon, welchen Wert er hat. Das ist nützlich, wenn wir nur wissen wollen, ob eine Information erfasst wurde.

Aufgaben

Schreibt auf overpass-turbo.eu eine eigene Postpass-SQL-Abfrage, die folgende Frage beantwortet: *Welche Restaurants im Seefeld haben sowohl einen Namen als auch eine angegebene Küche (`cuisine`)?*

Vorlage:

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}
SELECT osm_id, tags->>'name' AS name, tags->>'cuisine' AS cuisine, geom
FROM postpass_pointpolygon
WHERE tags->>'amenity' = 'restaurant'
  AND ...  -- ergänzt hier die Bedingung für 'name' und 'cuisine'
  AND geom && ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
ORDER BY name
```

Bonusfrage: Wie viele Restaurants liefert eure Abfrage zurück? Vergleicht das Ergebnis mit der Gesamtzahl aller Restaurants aus Abfrage 2.

Beim Aufbau komplexer SQL-Abfragen hilft es, Filter schrittweise einzufügen: erst alle Objekte des gewünschten Typs abfragen, dann sukzessive Bedingungen ergänzen. So lässt sich bei unerwarteten Ergebnissen schnell eingrenzen, welche Bedingung das Problem verursacht.

Lerneinheit 5 – Qualitätsprüfung als Datenbankanwendung

Dauer: ca. 35 Minuten

Lernziele

- Datenqualität als analytische Frage formulieren und mit SQL operationalisieren
- Qualitätskriterien wie Vollständigkeit, Konsistenz und Dubletten an OSM-Daten anwenden
- Einen strukturierten Qualitätsbericht für ein reales Quartier erstellen

Was ist Datenqualität?

Datenqualität beschreibt, wie gut Daten ihren beabsichtigten Zweck erfüllen. Wir unterscheiden vier Dimensionen:

- Vollständigkeit: Sind alle relevanten Objekte erfasst? Fehlen wichtige Attribute wie `name` oder `cuisine`?
- Konsistenz: Werden gleiche Dinge gleich modelliert? Ist das Trottoir als eigene Linie oder nur als Attribut erfasst?

- Aktualität: Sind die Daten aktuell? Ein Restaurant mit `opening_hours` von 2015 kann längst falsch sein.
- Redundanz / Dubletten: Gibt es dasselbe Objekt mehrfach? Zum Beispiel einmal als Punkt und einmal als Fläche?

Prüfung 1: Restaurants ohne Namen

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}
SELECT osm_type, osm_id, tags, geom
FROM postpass_pointpolygon
WHERE tags->>'amenity' = 'restaurant'
  AND NOT tags ? 'name'
  AND geom && ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
```

`NOT tags ? 'name'` ist wahr, wenn der Schlüssel `name` im Tags-Objekt fehlt. Das ist ein strikter Vollständigkeitscheck: Kein leerer String, kein NULL – der Key existiert schlicht nicht.

Prüfung 2: Restaurants ohne Küchenangabe

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}
SELECT osm_id, tags->>'name' AS name, tags, geom
FROM postpass_pointpolygon
WHERE tags->>'amenity' = 'restaurant'
  AND NOT tags ? 'cuisine'
  AND geom && ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
```

Prüfung 3: Trottoir-Abdeckung und -Konsistenz

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}
SELECT
  osm_id,
  tags->>'footway'   AS footway,
  tags->>'highway'   AS highway,
  tags->>'sidewalk'  AS sidewalk,
  tags->>'surface'   AS belag,
  geom
FROM postpass_line
WHERE geom && ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
  AND (
    tags->>'footway' = 'sidewalk'
    OR tags ? 'sidewalk'
    OR tags->>'highway' = 'footway'
  )
```

Aus dem Ergebnis lässt sich ablesen, wie viele Trottoirs als eigene Linien modelliert sind und wie viele nur als Attribut an einer Strasse hängen. Diese Inkonsistenz erschwert Auswertungen erheblich, weil man beide Varianten immer parallel mitdenken muss.

Prüfung 4: Qualitätsübersicht mit Aggregation

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}
SELECT
  kategorie,
  COUNT(*)                              AS total,
  COUNT(*) FILTER (WHERE mit_name)      AS mit_name,
  COUNT(*) FILTER (WHERE NOT mit_name)  AS ohne_name
FROM (
  SELECT
    'Street Art'   AS kategorie,
    tags ? 'name'  AS mit_name
  FROM postpass_point
  WHERE tags->>'tourism' = 'artwork'
    AND geom && ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
  UNION ALL
  SELECT
    tags->>'amenity' AS kategorie,
    tags ? 'name'    AS mit_name
  FROM postpass_pointpolygon
  WHERE tags->>'amenity' IN ('restaurant', 'cafe')
    AND geom && ST_MakeEnvelope(8.5469, 47.3492, 8.5721, 47.3599, 4326)
) sub
GROUP BY kategorie
ORDER BY total DESC
```

`COUNT(*) FILTER (WHERE …)` ist eine PostgreSQL-Kurzschreibweise für eine bedingte Zählung: Es wird nur gezählt, wenn die Bedingung wahr ist. Das Ergebnis gibt auf einen Blick einen Qualitätsüberblick über alle Kategorien.

Vorlage: Qualitätsbericht Quartier

Am Ende dieser Einheit erstellt ihr einen strukturierten Qualitätsbericht. Füllt die folgende Vorlage mit den Ergebnissen eurer Abfragen aus und präsentiert eure Ergebnisse in fünf Minuten der Klasse.

Gebiet: [Quartier / Ort] Abfragedatum: [Datum]

1. Street Art

   ```
   Anzahl gefundener Objekte: ___
   Objekte ohne `name`: ___ (___%)
   Objekte ohne `artwork_type`: ___ (___%)
   Objekte ohne `artist_name`: ___ (___%)
   ```

2. Restaurants

```
Anzahl als Punkte: ___
als Flächen: ___
Ohne `name`: ___
Ohne `cuisine`: ___
Ohne `opening_hours`: ___
```

3. Trottoirs und Gehwege

```
Als eigene Linien erfasst: ___
Als Strassenattribut erfasst: ___
Ohne `surface`: ___
Beobachtete Inkonsistenzen: ___
```

4. Hauptprobleme (kurze Beschreibung der grössten Lücken)
5. Verbesserungsvorschläge (was würdet ihr in OSM ergänzen, und warum?)

Abschlussaufgabe

Führt die vier Qualitätsprüfungen für ein Quartier eurer Wahl durch. Passt dazu die `ST_MakeEnvelope`-Koordinaten entsprechend an. Beantwortet in eurer Präsentation:

1. Wie vollständig sind die Daten in eurem Quartier?
2. Welches Objekt ist am schlechtesten erfasst? Warum könnte das so sein?
3. Was würdet ihr als Nächstes in OSM verbessern?

Fehlende Tags sind nicht immer gleichbedeutend mit unbekannten Informationen – manchmal wurden sie schlicht noch nicht erfasst. Beim Interpretieren der Qualitätszahlen lohnt es sich, diesen Unterschied zu berücksichtigen und gezielt nachzuschauen, ob ein Objekt in der Realität tatsächlich keine Angabe trägt.

Lerneinheit 6 – Visualisieren (optional)

Dauer: ca. 30 Minuten

Lernziele

- Abfrageergebnisse aus Postpass/overpass-turbo als GeoJSON oder CSV exportieren
- Geodaten in einem geeigneten Werkzeug visualisieren und kartografisch darstellen

Zur Erinnerung: Dieses Tutorial setzt voraus, dass man Grundlagen der Geovisualisierung kennt. Für diejenigen, die Grundlagen der Geovisualisierung

> kennenlernen möchten, gibt es einen Hinweis im Kapitel Voraussetzungen oben.

Daten exportieren

overpass-turbo bietet nach einer erfolgreichen Abfrage verschiedene Exportmöglichkeiten. Über den Button Export oben links lässt sich das Ergebnis als GeoJSON herunterladen. GeoJSON ist ein offenes Format für geografische Daten, das von den meisten Visualisierungswerkzeugen unterstützt wird.

Für tabellarische Auswertungen (z.B. Qualitätsberichte) können die Ergebnisse aus der Tabellenansicht von overpass-turbo als CSV kopiert oder direkt in eine Tabellenkalkulation übertragen werden.

Visualisieren mit uMap

uMap ist ein kostenloses Online-Werkzeug zum Erstellen von Karten auf Basis von OSM. Es erlaubt den direkten Import von GeoJSON-Dateien.

Vorgehensweise:

1. Führt eine Postpass-Abfrage auf overpass-turbo.eu aus (z.B. alle Street-Art-Objekte im Seefeld).
2. Exportiert das Ergebnis als GeoJSON (Export → GeoJSON).
3. Öffnet umap.openstreetmap.fr und erstellt eine neue Karte.
4. Importiert die GeoJSON-Datei über Daten importieren.
5. Passt Farben, Symbole und Popups nach Bedarf an.

Visualisieren mit QGIS (alternativ)

QGIS ist ein kostenloses Desktop-GIS-Programm für komplexere Visualisierungen und räumliche Analysen.

1. Öffnet QGIS und wählt Layer → Layer hinzufügen → Vektorlayer.
2. Ladet die exportierte GeoJSON-Datei.
3. Über Layereigenschaften → Darstellung können Symbole und Farben angepasst werden.
4. Mit dem Drucklayout lässt sich eine druckfertige Karte erstellen.

Aufgaben

Visualisiert die Ergebnisse aus Lerneinheit 5 auf einer Karte:

1. Exportiert alle Street-Art-Objekte im Seefeld als GeoJSON.
2. Importiert die Daten in uMap oder QGIS.
3. Färbt die Objekte nach einem Qualitätsmerkmal ein (z.B. grün = hat `artist_name`, rot = fehlt `artist_name`).
4. Welche räumlichen Muster erkennt ihr? Gibt es Strassenzüge oder Quartierteile, die besser oder

schlechter erfasst sind?

Für eine erste Exploration genügen geojson.io oder uMap völlig. QGIS lohnt sich, wenn Symbole nach mehreren Attributen gleichzeitig gesteuert oder druckfertige Karten erstellt werden sollen.

Abschluss: Was haben wir gelernt und wie geht es weiter?

Was wir gelernt haben

In diesem Workshop haben wir OpenStreetMap nicht nur als Karte, sondern als Datenbank kennengelernt. Wir haben gesehen, wie reale Objekte – Street Art, Restaurants, Trottoirs – als Nodes, Ways und Tags modelliert werden, und warum diese flexible Struktur sowohl eine Stärke als auch eine Herausforderung ist. Mit SQL und Postpass konnten wir gezielt Abfragen stellen, Qualitätsprobleme aufdecken und die Ergebnisse visualisieren.

Die wichtigsten Erkenntnisse: Daten entstehen nicht einfach – sie werden von Menschen erfasst, und das sieht man ihnen an. Vollständigkeit, Konsistenz und Aktualität sind keine Selbstverständlichkeit, sondern das Ergebnis kollektiver Arbeit. Und: Mit den richtigen Werkzeugen lassen sich auch grosse, unstrukturierte Datensätze systematisch analysieren.

Wie geht es weiter?

Bei Street Art bzw. Kunst im öffentlichen Raum, die in OSM erfasst ist, wäre es schön, wenn man zu den Objekten auch Fotos und Bilder verwalten kann. In OSM selbst werden keine Bilder gespeichert, aber OSM kann auf Bilder verweisen. Solche Bilder können zum Beispiel mit der Commons App in Wikimedia Commons hochgeladen werden, sofern die Urheberrechte und Lizenzen dies erlauben. In OSM kann man danach mit dem Tag `wikimedia_commons=*` auf die passende Commons-Kategorie oder Datei verlinken, zum Beispiel `wikimedia_commons=Category:Street art in Zürich` oder `wikimedia_commons=File:Beispielbild.jpg`. Für einzelne direkte Bild-URLs wird zusätzlich häufig `image=_` verwendet (wiki.openstreetmap.org).

Wer selbst zur Verbesserung der OSM-Daten beitragen möchte, kann an einem Mapathon teilnehmen. Bei einem Mapathon arbeiten Gruppen gemeinsam daran, ein bestimmtes Gebiet in OSM zu vervollständigen – zum Beispiel fehlende Street-Art-Objekte, Restaurants ohne `cuisine`-Tag oder Trottoirs ohne Belagsangabe zu ergänzen. Mapathons werden regelmässig von lokalen OSM-Communities, Hochschulen und NGOs organisiert.

Ein guter Einstieg ist die Webseite openstreetmap.org/about, wo lokale Gruppen und Veranstaltungen aufgelistet sind wie zum Beispiel die OSM.ch. Wer lieber im eigenen Quartier kartieren möchte, kann direkt den im OSM.org eingebauten Editor (iD-Editor) nutzen oder zum Beispiel die mobile App StreetComplete (Android) oder die mobile Webapp MapComplete verwenden, die beide gezielte Aufgaben zum Ergänzen fehlender Tags anbieten.

Weitere Tutorials und Lernmaterialien

LearnOSM (learnosm.org) ist das offizielle Lernportal der OSM-Community und erklärt Schritt für Schritt, wie man Daten erfasst, den JOSM-Editor bedient und an Mapping-Projekten teilnimmt.

Das OSM-Wiki (wiki.openstreetmap.org) ist die umfassendste Referenz für alle Tagging-Konventionen und eignet sich, um nachzuschauen, wie ein bestimmtes Objekt korrekt erfasst wird.

OpenSchoolMaps (openschoolmaps.ch) bietet eine Sammlung von Unterrichtsmaterialien rund um OSM und Geodaten, speziell für den Schulunterricht aufbereitet. Die Themen reichen von der einfachen Kartennutzung von OpenStreetMap über die Datenanalyse mit QGIS bis zu weiteren Informatikthemen.

Auf OpenSchoolMaps gibt es ein spezielles Tutorial für das OpenStreetMap-Daten extrahieren mit SQL und Postpass. Und es lohnt sich ein Blick auf die Postpass-Beispielsammlung im OSM-Wiki (wiki.openstreetmap.org/wiki/Postpass/Examples).

Glossar

Bounding Box

Ein rechteckiger geografischer Ausschnitt, definiert durch minimale und maximale Längen- und Breitengrade. Wird unter anderem in PostGIS-Abfragen verwendet, um Ergebnisse auf ein bestimmtes Gebiet einzuschränken. In PostGIS erzeugt `ST_MakeEnvelope(west, süd, ost, nord, 4326)` eine solche Bounding Box als Geometrie.

Entität

Ein eindeutig identifizierbares, abgrenzbares Objekt der realen Welt, über das Daten gespeichert werden sollen. Beispiele: ein bestimmtes Street-Art-Objekt, ein konkretes Restaurant.

JSONB

Ein Datentyp in PostgreSQL für JSON-Daten in binärer, indizierbarer Form. Ermöglicht effiziente Abfragen auf einzelne Schlüssel mit dem Operator `->>'key'` (liefert den Wert als Text) und Existenzprüfungen mit `? 'key'` (prüft, ob ein Schlüssel vorhanden ist).

Node / Way / Relation

Die drei Grundobjekttypen in OpenStreetMap. Ein Node ist ein Punkt mit Koordinaten. Ein Way ist eine geordnete Liste von Nodes und kann eine offene Linie oder eine geschlossene Fläche bilden. Eine Relation gruppiert mehrere Objekte mit einer definierten Rolle (z.B. eine Busroute oder ein Stadtquartier).

Normalisierung

Ein Prinzip des Datenbankdesigns, bei dem redundante Daten eliminiert werden, indem zusammengehörende Informationen in separate Tabellen ausgelagert werden. Ziel: keine unnötigen Wiederholungen und konsistente Daten ohne Anomalien bei Änderungen.

PostGIS

Eine Erweiterung für PostgreSQL, die räumliche Datentypen und Funktionen bereitstellt.

Ermöglicht das Speichern und Abfragen von Koordinaten, Linien und Flächen direkt in SQL.

Postpass

Ein Dienst von Geofabrik, der aktuelle OSM-Daten in eine PostgreSQL/PostGIS-Datenbank importiert und über eine SQL-Schnittstelle zugänglich macht. Auf overpass-turbo.eu wird Postpass durch die Kopfzeile `{{data:sql,server=https://postpass.geofabrik.de/api/0.2/}}` aktiviert.

Schema-on-read

Ein Designprinzip, bei dem die Struktur der Daten nicht vor dem Einfügen definiert wird, sondern erst beim Lesen interpretiert wird. OSM-Tags folgen diesem Prinzip: Ein Node mit `amenity=restaurant` wird zum Restaurant durch Interpretation, nicht durch seine Tabellenspalte.

Schema-on-write

Das klassische Prinzip relationaler Datenbanken: Die Tabellenstruktur (Schema) wird vor der Dateneingabe definiert, und alle Einträge müssen sich daran halten.

SRID

Spatial Reference ID – eine Zahl, die ein Koordinatensystem eindeutig identifiziert. Der SRID 4326 steht für WGS 84, das Koordinatensystem von GPS und OpenStreetMap, das Längen- und Breitengrade in Dezimalgrad verwendet.

Tag

Ein Schlüssel-Wert-Paar in OpenStreetMap der Form `key=value`, das die Eigenschaften eines Objekts beschreibt. Beispiele: `amenity=restaurant`, `name=Kronenhalle`, `tourism=artwork`. Alle Tags zusammen ergeben das vollständige Profil eines OSM-Objekts.

Noch Fragen? Wende dich an OpenStreetMap Schweiz oder Stefan Keller!