OpenSchoolMaps: Ein sanfter Einstieg in Spatial SQL mit PostGIS Lösungen

OpenSchoolMaps.ch — Freie Lernmaterialien zu freien Geodaten und Karten

Stefan Keller und Jonas Grüter, Rapperswil, 2026-06-12

Einführung

Dieses Tutorial bietet einen sanften Einstieg in **Spatial SQL**: sanft, weil SQL-Kenntnisse genügen und kein weiteres Vorwissen nötig ist, weil die Konzepte schrittweise eingeführt werden, und weil alle Übungen ohne Installation direkt im Browser laufen.

Nach dem Durcharbeiten kennt man die **wichtigsten Konzepte von Spatial SQL**: vom Erkunden eines räumlichen Datenbankschemas über Messfunktionen und räumliche Prädikate bis hin zu Spatial Joins und Koordinatenreferenzsystemen.

Wer `SELECT` … `FROM` … `WHERE` kennt, kennt bereits das wichtigste Werkzeug. Spatial SQL erweitert dieses um **geometrische Datentypen und Funktionen**. Damit kann man nicht nur die Frage ”Welche Kunden haben mehr als 1000 CHF ausgegeben?” stellen, sondern auch ”Welche Kunden wohnen innerhalb von 500 m von einer Filiale?”.

Bildquelle: Generiert mit Gemini

Zielgruppe sind GIS-Interessierte mit SQL-Grundkenntnissen.

Voraussetzungen: Grundlegende Kenntnisse von SQL, insbesondere SQL-Abfragen.

> Wer sich die wichtigsten Konzepte von SQL erarbeiten möchte, dem seien folgende Tutorials empfohlen: "W3C Schools", "SQLBolt" "Eskuel Suite infs.ch" (SQL Game

> Console → open game → SQL Island) oder SQL-Island.

Dieses Tutorial basiert auf dem bekannten Workshop "Introduction to PostGIS" (siehe Kapitel "Abschluss" unten) inklusive Übungsdaten von New York City 2020 (sogenannte **"NYC-Übungsdatenbank"**).

Alle **Übungen** laufen direkt im Browser, ohne Installation, dank der interaktiven, browserbasierten Umgebung (Playground) "SQLize": https://sqlize.online/s/pQ (PostgreSQL 17 + PostGIS 3.6, Übungsdaten bereits geladen, readonly).

> Mit dem "Postgres Playground" von CrunchyData gibt es eine Alternative zu SQLize. Dabei handelt es sich um eine interaktive, browserbasierte Umgebung, die ebenfalls Daten von New York City geladen hat.

PostGIS ist eine Open-Source-Erweiterung für die Datenbank PostgreSQL. "GIS" steht für Geoinformationssysteme. PostGIS ermöglicht die Verwendung räumlicher Datentypen (Punkte, Linien und Polygone) sowie von über 500 Funktionen nach dem Standard des Open Geospatial Consortium (OGC) in einer relationalen Datenbank. Spatial SQL ist heute keine Nischentechnologie mehr, denn neben PostGIS unterstützen auch andere Systeme räumliche Abfragen, etwa **DuckDB**. PostGIS ist jedoch praktisch die Referenzimplementierung für Spatial SQL. Für dieses Tutorial gibt es einen wichtigen praktischen Grund: Mit SQLize steht ein frei verfügbares PostGIS mit vorbereitetem Übungsdatensatz bereit, das sofort einsatzfähig ist.

Das Tutorial besteht aus den folgenden Lerneinheiten und dauert insgesamt 2.5 Stunden.

1. Den Datensatz erkunden (30 min)
2. Geometrien und Messfunktionen (30 min)
3. Räumliche Beziehungen (30 min)
4. Räumliche Verknüpfungen (Spatial Joins) (30 min)
5. Koordinatenreferenzsysteme & Projektion (30 min)

Konventionen

Zur besseren Lesbarkeit gelten folgende typografische Konventionen:

- Fett kennzeichnet Anweisungen und Funktionsnamen beim ersten Auftreten, z. B. **ST_Area(geom)**.
- `Festbreitenschrift` steht für Dateinamen, Tabellen- und Spaltennamen, z. B. `nyc_streets`.
- SQL-Abfragen erscheinen stets in einem Blocksatz-Fliesstext.

> SQL-Schlüsselwörter werden hier aus didaktischen Gründen in Grossbuchstaben geschrieben. In der Praxis empfiehlt es sich jedoch, Schlüsselwörter in **Kleinbuchstaben** zu schreiben (ebenso wie Tabellen- und Attributnamen, zumindest in PostgreSQL).

Räumliche Datentypen

PostGIS führt den Datentyp `GEOMETRY` ein, der alle gängigen Vektorgeometrien abbildet:

Point
: ein einzelner Koordinatenpunkt, z. B. eine U-Bahn-Station.

LineString
: eine geordnete Folge von Punkten, z. B. ein Strassenabschnitt.

Polygon
: eine geschlossene Fläche aus einem Aussenring und optionalen Innenringen (Löcher), z. B. ein Stadtteil.

MultiPoint
: eine Sammlung von Punkten als einzelnes Geometrieobjekt.

MultiLineString
: eine Sammlung von LineStrings, z. B. ein Strassenzug mit Unterbrechungen.

MultiPolygon
: eine Sammlung von Polygonen, z. B. ein Borough mit Inseln.

GeometryCollection
: eine heterogene Sammlung beliebiger Geometrietypen.

Die räumlichen Funktionen von PostGIS lassen sich einer der folgenden fünf Kategorien zuordnen:

1. Konvertierung — Funktionen, die Geometrien und externe Datenformate konvertieren (z. B. `ST_AsText`, `ST_AsGeoJSON`).
2. Verwaltung — Funktionen, die Informationen über räumliche Tabellen verwalten (z. B. `UpdateGeometrySRID`).
3. Abfrage — Funktionen, die Eigenschaften und Masse einer Geometrie abfragen (z. B. `ST_Area`, `ST_Length`, `ST_GeometryType`).
4. Vergleich — Funktionen, die zwei Geometrien hinsichtlich ihrer räumlichen Beziehung vergleichen (z. B. `ST_Intersects`, `ST_Contains`, `ST_DWithin`).
5. Erzeugung — Funktionen, die aus anderen Geometrien neue Geometrien erzeugen (z. B. `ST_Buffer`, `ST_Union`, `ST_Transform`).

Ein Wort zu SQL

In diesem Tutorial werden ausschliesslich SELECT-Anweisungen behandelt. Die Übungsdaten sind ja schliesslich nur readonly. Die SQL-Anweisungen zum Einfügen, Aktualisieren oder Löschen von Daten (`INSERT`, `UPDATE` und `DELETE`) sind weniger kompliziert.

SQL ist die wohl bekannteste deklarative Computersprache. Sie funktioniert grundlegend anders als Sprachen wie Python oder Java. In SQL wird lediglich deklariert, welche Ergebnisse erzielt werden

sollen, nicht jedoch, wie der Computer diese berechnet. Das ist elegant und ermöglicht es einer Datenbank zudem, Abfragen für eine bessere Performance zu optimieren. Das heisst: Schrittweise Anweisungen sollten vergessen werden. Stattdessen sollte darüber nachgedacht werden, wie Sachverhalte deklariert werden können, und nicht darüber, was in welcher Reihenfolge berechnet werden muss.

Ausserdem sollte stets die logische Reihenfolge der SQL-Klauseln im Hinterkopf behalten werden, um häufige Fehler zu vermeiden. Denn die SQL-Syntax ist nicht "geordnet". Die Syntaxschlüsselwörter erscheinen nicht in der Reihenfolge, in der sie ausgeführt werden. Tatsächlich kommt das SELECT-Schlüsselwort fast am Ende der Ausführungsreihenfolge. Vergleichen wir die beiden Ordnungen:

Die syntaktische Reihenfolge von SQL lautet: `SELECT FROM WHERE GROUP BY UNION ORDER BY`. Der Einfachheit halber sind hier nicht alle SQL-Klauseln aufgeführt. Diese Reihenfolge unterscheidet sich grundlegend von der logischen Reihenfolge: `FROM WHERE GROUP BY SELECT UNION ORDER BY`.

Dabei sind folgende Punkte zu beachten:

1. `FROM` ist die erste Klausel, nicht `SELECT`. Als Erstes werden Daten von der Festplatte in den Arbeitsspeicher geladen, um damit zu arbeiten.
2. `SELECT` wird nach den meisten anderen Klauseln ausgeführt – insbesondere nach `FROM` und `GROUP BY`. Dies ist wichtig zu verstehen, wenn man glaubt, in der `WHERE`-Klausel auf Ausdrücke verweisen zu können, die in der `SELECT`-Klausel definiert wurden. Folgendes ist zum Beispiel *nicht* möglich:

```
SELECT A.x + A.y AS z
FROM A
WHERE z = 10 -- z ist hier noch nicht verfügbar!
```

Wenn `z` wiederverwendet werden soll, gibt es zwei Möglichkeiten. Entweder wird der Ausdruck wie nachfolgend skizziert wiederholt – oder es werden Unterabfragen, Common Table Expressions oder Views verwendet, um Code-Wiederholungen zu vermeiden.

```
SELECT A.x + A.y AS z
FROM A
WHERE (A.x + A.y) = 10
```

Bei Abfragen lohnt es sich immer, mit `ORDER BY` zu arbeiten, vor allem, wenn die Datenmenge unbekannt ist.

Lerneinheit 1 — Die Datenbank erkunden

Lernziel: Die Teilnehmer sollen verstehen, welche Tabellen im Playground vorhanden sind, was sie enthalten und wie sie zusammenhängen.

Zeitbedarf: ca. 30 min (Instruktion ~10 min, Übungen ~20 min).

Bevor man eine fremde Stadt erkundet, schaut man sich den Stadtplan an. Welche Stadtteile gibt es? Wo liegen die U-Bahn-Stationen? Dasselbe macht man mit einem neuen Datenbankschema — erst muss man verstehen, was vorhanden ist, dann kann man es analysieren.

Die vier Tabellen

Die NYC-Übungsdatenbank besteht aus vier räumlichen Tabellen. Alle Geometrien liegen im Koordinatensystem UTM Zone 18N (EPSG:26918). Ein vollständigeres Datenschema ist im Appendix dokumentiert.

Die Tabelle `nyc_census_blocks` — Volkszählungsblöcke (38794 Datensätze). Ein Census Block ist die kleinste Einheit, für die Volkszählungsdaten erfasst werden.

- `popn_total` — Gesamtbevölkerung des Blocks
- `boroname` — Name des Boroughs
- `geom` — Polygongeometrie des Blocks

Tabelle `nyc_neighborhoods` — Stadtteile (129 Datensätze). Stadtteile sind soziale Konstrukte ohne feste, amtliche festgelegte Grenzen.

- `name` — Name des Stadtteils
- `boroname` — Name des Boroughs
- `geom` — Polygongeometrie des Stadtteils

Tabelle `nyc_streets` — Strassennetz (19091 Datensätze). Das Strassenmittelliniennetz der Stadt.

- `name` — Strassenname
- `type` — Strassentyp: primary, secondary, residential, motorway, ...
- `geom` — Liniengeometrie (MultiLineString)

Tabelle `nyc_subway_stations` — U-Bahn-Stationen (491 Datensätze).

- `name` — Stationsname
- `borough` — Borough der Station
- `routes` — Linien, die durch die Station fahren, z.B. "A,C,E"
- `express` — Haelt ein Expresszug? "express" = ja, "" = nein
- `geom` — Punktgeometrie der Station

Schema-Metadaten abfragen

PostGIS verfügt über eine eigene Metadatentabelle bereit, die alle räumlichen Spalten der Datenbank auflistet:

```
SELECT f_table_name AS table,
       f_table_scheme AS scheme,
```

```
       type         AS geometry_type,
       srid
FROM geometry_columns
WHERE f_table_scheme IN ('public','postgis')
ORDER BY f_table_name;
```

Übungen

Die Übungen sind jeweils wie folgt strukturiert: Zunächst gibt es eine Aufgabenstellung, die typischerweise eine SQL-Abfrage verlangt, gefolgt von einem Tipp. Im Falle von SQL-Abfragen folgt dann "Abfrageergebnis anzeigen", d. h. eine Beschreibung des sogenannten Result Sets (oder die resultierende Tabelle selbst) nach Ausführung der korrekten Abfrage. Schliesslich kann man sich auch die "Lösung anzeigen" lassen.

Übung 1.1

Welche Tabellen sind im Playground vorhanden? Frage dazu die Metadatentabelle `geometry_columns` ab: Welche Geometrietypen und SRIDs sind vorhanden?

`f_table_schema = 'public'` werden eigene Tabellen eingeschränkt und interne PostGIS-Systemtabellen ausgeblendet.

Abfrageergebnis

Alle vier räumlichen Tabellen erscheinen mit dem Typ (MULTIPOLYGON, MULTILINESTRING oder POINT) und dem SRID 26918.

Lösung

```
SELECT f_table_name AS table,
       type         AS geometry_type,
       srid
FROM geometry_columns
WHERE f_table_scheme IN ('public','postgis')
ORDER BY f_table_name;
```

Übung 1.2

Wie sehen die ersten drei Zeilen der Tabelle `nyc_subway_stations` aus und welche Spalten enthält sie?

`SELECT * ⋯ LIMIT 3` gibt alle Spalten zurück und begrenzt die Ausgabe auf drei Zeilen — das ist nützlich,um sich einen ersten Überblick über unbekannte Tabellen zu verschaffen.

Abfrageergebnis

Die Spalte `geom` enthält Binärdaten (WKB). Mit der Funktion `ST_AsText(geom)` lässt sie sich lesbar machen — das folgt in LE 2.

Lösung

```
SELECT * FROM nyc_subway_stations LIMIT 3;
```

Übung 1.3

Wie viele Datensätze enthält jede der vier räumlichen Tabellen? Gib das Ergebnis als eine einzige Tabelle aus.

`UNION ALL` fügt die Ergebnisse mehrerer `SELECT`-Abfragen zeilenweise zusammen, ohne Duplikate zu entfernen.

Abfrageergebnis

Ergebnisse: 38794 / 129 / 19091 / 491

Lösung

```
SELECT 'nyc_census_blocks'   AS tabelle, Count(*) AS amount FROM nyc_census_blocks
UNION ALL
SELECT 'nyc_neighborhoods',              Count(*)           FROM nyc_neighborhoods
UNION ALL
SELECT 'nyc_streets',                    Count(*)           FROM nyc_streets
UNION ALL
SELECT 'nyc_subway_stations',            Count(*)           FROM
nyc_subway_stations;
```

Übung 1.4

Welche verschiedenen Strassentypen (`type`) gibt es in `nyc_streets`, und wie häufig kommt jeder Typ vor? Sortiere die Ergebnisse absteigend nach Häufigkeit.

`GROUP BY type` fasst Zeilen mit gleichem Wert zusammen; `Count(*)` zählt die Einträge je Gruppe.

Abfrageergebnis

Am häufigsten kommt der Typ "residential" vor, gefolgt von "motorway_link" und "footway".

Lösung

```
SELECT type, Count(*) AS amount
FROM nyc_streets
GROUP BY type
ORDER BY amount DESC;
```

Übung 1.5

Welche U-Bahn-Linien sind im Datensatz enthalten? Gib jede Linie nur einmal aus.

Die Spalte `routes` enthält kommagetrennte Linienkürzel wie "A,C,E". Mit `DISTINCT` werden Duplikate entfernt; Eine vollständige Zerlegung in Einzellinien folgt in späteren LE.

Abfrageergebnis

Es gibt Stationen mit Einzellinien ("1", "2") und Umsteigeknoten mit mehreren Linien ("A,C", "4,5,6", ...).

Lösung

```
SELECT DISTINCT routes
FROM nyc_subway_stations
ORDER BY routes;
```

Übung 1.6 (Bonus)

Wie viele Expressstationen gibt es pro Borough? Sortiere absteigend.

Die Spalte `express` enthält den Wert "express" für Expressstationen und einen leeren String für normale Stationen.

Abfrageergebnis

Mit Abstand hat Manhattan die meisten Expressstationen.

Lösung

```
SELECT borough, Count(*) AS express_stations
FROM nyc_subway_stations
WHERE express = 'express'
GROUP BY borough
ORDER BY express_stations DESC;
```

Die Metadatentabelle `geometry_columns` ist Teil des OGC-Standards "Simple Features for SQL". PostGIS führt daneben auch die Tabelle `geography_columns` für den Geography-Typ (Kugelkoordinaten). Wer verstehen will, wie PostGIS räumliche Indizes verwaltet, findet in der PostgreSQL-Dokumentation unter "GIST Index" den Einstieg.

Lerneinheit 2 — Geometrien und Messfunktionen

Lernziel: Die verschiedenen Geometrietypen kennenlernen und räumliche Eigenschaften (Fläche, Länge, Koordinaten) abfragen können.

Zeitbedarf: ca. 30 min (Instruktion ~10 min, Übungen ~20 min).

PostGIS-Funktionen beginnen immer mit `ST_` (*Spatial Type*). `ST_Area(geom)` fragt: "Wie gross ist diese Fläche?" — genauso, wie eine Karten-App die Fläche eines markierten Gebiets anzeigen kann.

Wichtige Messfunktionen

Funktion	Beschreibung	Einheit
`ST_GeometryType(geom)`	Typ: Point, LineString, Polygon, …	Text
`ST_Area(geom)`	Fläche	m² (bei UTM-Daten)
`ST_Length(geom)`	Länge einer Linie	m (bei UTM-Daten)
`ST_X(geom) / ST_Y(geom)`	X/Y-Koordinate eines Punktes	Karteneinheit
`ST_AsText(geom)`	Geometrie als WKT-Text ausgeben	Text
`ST_AsGeoJSON(geom)`	Geometrie als GeoJSON	JSON

Die NYC-Daten liegen in UTM Zone 18N (EPSG:26918) — alle Längen sind in Metern angegeben, die Flächen in m². Um km zu erhalten, teilen Sie durch 1000, um Hektar zu erahalten, teilen Sie durch 10000.

Übungen

Übung 2.1

Wie gross ist der Stadtteil "West Village" in Hektar?

1 Hektar = 10000 m². Für Acres durch 4047 teilen.

Abfrageergebnis

Ergebnis: ~104.5 ha

Lösung

```
SELECT ST_Area(geom) / 10000 AS area_ha
FROM nyc_neighborhoods
WHERE name = 'West Village';
```

Übung 2.2

Welchen Geometrietyp hat die Strasse "Pelham St", und wie lang ist sie in Metern?

Die Funktion `ST_GeometryType` und `ST_Length` lassen sich in der gleichen `SELECT`-Zeile kombinieren.

Abfrageergebnis

Ergebnis: `ST_MultiLineString`, ~50.3 m

Lösung

```
SELECT ST_GeometryType(geom), ST_Length(geom)
FROM nyc_streets
WHERE name = 'Pelham St';
```

Übung 2.3

Wie lang ist das gesamte Strassennetz von NYC in km? Summiere dazu alle Strassensegmente.

Die Funktion `SUM(ST_Length(geom))` addiert die Länge aller Geometrien in der Tabelle. Das Ergebnis ist in Metern (UTM). Durch 1000 teilen, um km zu erhalten.

Abfrageergebnis

Ergebnis: ~10419 km

Lösung

```
SELECT SUM(ST_Length(geom)) / 1000 AS lenght_km
FROM nyc_streets;
```

Übung 2.4

Welche U-Bahnstation liegt am weitesten westlich?

Westlich = kleinster X-Wert in UTM. `ORDER BY ⋯ LIMIT 1` gibt nur die erste Zeile zurück.

Abfrageergebnis

Ergebnis: Tottenville

Lösung

```
SELECT name, ST_X(geom) AS x_coordinate
FROM nyc_subway_stations
ORDER BY ST_X(geom)
LIMIT 1;
```

Übung 2.5 (Bonus)

Wie gross ist Manhattan in Hektar? Summiere die Flächen aller Stadtteile im Borough Manhattan.

Mit `boroname = 'Manhattan'` wird in `nyc_neighborhoods` nach dem Borough gefiltert. Da die Tabelle jedoch nur die Stadtteile und nicht die einzelnen Blocks enthält, ist die Summe der Stadtteilflächen nur eine Näherung.

Abfrageergebnis

Ergebnis: ~5651 ha

Lösung

```
SELECT SUM(ST_Area(geom)) / 10000 AS area_ha
```

```
FROM nyc_neighborhoods
WHERE boroname = 'Manhattan';
```

`ST_AsText` gibt das menschenlesbare Well-Known Text (WKT)-Format zurück. Für die Übertragung in Web-Anwendungen ist `ST_AsGeoJSON` oft besser geeignet. Wer Geometrien nicht nur lesen, sondern auch konstruieren möchte, findet mit `ST_MakePoint(x, y)` und `ST_GeomFromText(wkt, srid)` zwei wichtige Einstiegsfunktionen.

Lerneinheit 3 — Räumliche Beziehungen

Lernziel: Anwendung räumlicher Räumliche Prädikate (enthält, schneidet, Abstandsfilter) auf echte Geodaten.

Zeitbedarf: ca. 30 min (Instruktion ~10 min, Übungen ~20 min).

`ST_Intersects` ist wie die Frage "Überlappt sich dieses Gebiet mit jenem?" auf einer Karte. `ST_DWithin` ist der Umkreisfilter in einer Karten-App: "Zeige alle Restaurants innerhalb von 500 m von meinem Standort."

Räumliche Prädikate

Diese Funktionen geben `true` oder `false` zurück und eignen sich direkt für das `WHERE`:

- **`ST_Intersects(A, B)`** — A und B überschneiden sich (häufigste Funktion).
- **`ST_Contains(A, B)`** — A enthält B vollständig.
- **`ST_Within(A, B)`** — A liegt vollständig in B (Umkehrung von Contains).
- **`ST_DWithin(A, B, r)`** — A liegt innerhalb von r Metern von B.

Mit `ST_GeomFromText('LINESTRING(…)', 26918)` lässt sich eine Geometrie direkt im SQL angeben. Der zweite Parameter ist der *SRID* (Spatial Reference ID, hier UTM 18N).

Mit der Funktion `ST_GeomFromText` ohne Angabe des SRID wird eine Geometrie mit SRID 0 erzeugt. In diesem Fall funktionieren räumliche Abfragen nicht korrekt. Immer `26918` mitgeben!

Übungen

Übung 3.1

In welchem Stadtteil und Borough liegt die Strasse "Atlantic Commons"?

Diese Aufgabe erfordert zwei Schritte: Zunächst muss die Geometrie der Strasse aus `nyc_streets` abgefragt werden, anschliessend muss mit der Funktion `ST_Intersects` der Stadtteil in der

Tabelle `nyc_neighborhoods` gefunden werden.

Den WKT-Text aus Schritt 1 als `LINESTRING`-Literal in Schritt 2 einfügen — `ST_GeomFromText('LINESTRING(…)', 26918)`. Der SRID 26918 muss mit dem der Tabelle übereinstimmen.

Abfrageergebnis

- Schritt 1 Ergebnis: `MULTILINESTRING586782 4504202, 586864 4504216`
- Schritt 2 Ergebnis: Fort Greene, Brooklyn

Lösung

Schritt 1 — Geometrie der Strasse abfragen:

```
SELECT ST_AsText(geom)
FROM nyc_streets
WHERE name = 'Atlantic Commons';
```

Schritt 2 — Stadtteil per räumlicher Überschneidung finden:

```
SELECT name, boroname
FROM nyc_neighborhoods
WHERE ST_Intersects(
  geom,
  ST_GeomFromText('LINESTRING(586782 4504202, 586864 4504216)', 26918)
);
```

Übung 3.2

Wie viele Menschen wohnen in einem Umkreis von 50 m um die "Atlantic Commons"?

Der dritte Parameter von `ST_DWithin` ist der Radius in Metern.

Abfrageergebnis

Ergebnis: 1438 Personen

Lösung

```
SELECT SUM(popn_total)
FROM nyc_census_blocks
WHERE ST_DWithin(
  geom,
```

```
  ST_GeomFromText('LINESTRING(586782 4504202, 586864 4504216)', 26918),
  50
);
```

Wenn man `ST_DWithin` mit Standard-GPS-Koordinaten (Geometrie 4326) verwendet, wird die Entfernung in Grad statt in Metern gemessen. Das ist kaum, was man will. Um die Entfernung in Metern zu erhalten, wandelt man die Daten in den Typ "geography" um, indem man "::geography" hinter den Datenpunkten hinzufügt, wie hier: `ST_DWithin(your_column::geography, your_point::geography, 1000)`.

Übung 3.3 (Bonus)

Welche Strassen stossen an die "Atlantic Commons" an (Abstand < 0.1 m)?

Ein Abstand von 0.1 m entspricht praktisch einer direkten Berührung, was wegen minimaler Koordinatenungenauigkeiten nötig ist.

Abfrageergebnis

Ergebnis: Atlantic Commons, Cumberland St

Lösung

```
SELECT name
FROM nyc_streets
WHERE ST_DWithin(
  geom,
  ST_GeomFromText('LINESTRING(586782 4504202, 586864 4504216)', 26918),
  0.1
);
```

`ST_Intersects` nutzt intern den räumlichen GiST-Index und ist dadurch auch bei grossen Tabellen schnell. Wer die zugrunde liegenden DE-9IM-Topologieregeln verstehen möchte (warum `ST_Touches` vs. `ST_Overlaps`), findet im PostGIS-Workshop-Kapitel "Spatial Relationships" eine gute Erklärung.

Lerneinheit 4 — Räumliche Verknüpfungen (Spatial Joins)

Lernziel: Zwei Tabellen anhand ihrer räumlichen Lage miteinander verknüpfen.

Zeitbedarf: ca. 30 min (Instruktion ~10 min, Übungen ~20 min).

Ein klassischer SQL-Join fragt: "Haben diese beiden Datensätze die gleiche ID?" Ein Spatial Join fragt dagegen: "Liegen diese beiden Objekte am gleichen Ort?" — Das ist vergleichbar mit der Prüfung, ob eine U-Bahn-Station in einem bestimmten Stadtteil liegt, wie man es auf einer Karte tun würde.

Grundstruktur

```
SELECT a.name, b.name
FROM tabelle_a AS a
JOIN tabelle_b AS b
  ON ST_Contains(a.geom, b.geom)
WHERE ...
```

Tabellenaliase (`AS a`, `AS b`) vermeiden Namenskonflikte bei gleichnamigen Spalten wie `name` oder `geom`. Die Funktion `DISTINCT` verhindert doppelte Ergebniszeilen, wenn mehrere Punkte in der gleichen Fläche liegen. `ST_Contains` verlangt das vollständige Enthaltensein. Bei Grenzobjekten ist `ST_Intersects` robuster.

`strpos(s.routes, '6') > 0` sucht die Ziffer "6" im Textfeld `routes` (z.B. `'4,5,6'`). `= '6'` würde dagegen nur Stationen mit exakt "6" finden.

Übungen

Übung 4.1

Welche U-Bahnstation liegt im Stadtteil "Little Italy", und auf welcher Linie fährt sie?

Verknüpfe `nyc_subway_stations` mit `nyc_neighborhoods` über `ST_Contains(n.geom, s.geom)` — der Stadtteil enthält den Stationspunkt.

Abfrageergebnis

Ergebnis: Spring St, Linie 6

Lösung

```
SELECT s.name, s.routes
FROM nyc_subway_stations AS s
JOIN nyc_neighborhoods AS n
  ON ST_Contains(n.geom, s.geom)
WHERE n.name = 'Little Italy';
```

Übung 4.2

Welche Stadtteile werden von der Linie 6 bedient?

Die Funktion `DISTINCT` entfernt Duplikate, da eine Station am Rand zweier Stadtteile würde sonst doppelt erscheinen würde.

Abfrageergebnis

Ergebnis: mehrere Stadtteile in Manhattan und der Bronx

Lösung

```
SELECT DISTINCT n.name, n.boroname
FROM nyc_subway_stations AS s
JOIN nyc_neighborhoods AS n
  ON ST_Contains(n.geom, s.geom)
WHERE strpos(s.routes, '6') > 0
ORDER BY 2,1;
```

Übung 4.3

Wie viele Einwohner lebten zum Zeitpunkt der Volkszählung im Stadtteil "Battery Park" (Beisielsweise als Schätzung für die Evakuierungszahl nach dem 11. September)?

Verwende statt `ST_Contains` die Funktion `ST_Intersects`, da die Census Blocks an der Stadtteilgrenze trotzdem gezählt werden sollen. Bei der Abfrage muss das Datum nicht berücksichtigt werden.

Abfrageergebnis

Ergebnis: ~17153 Personen

Lösung

```
SELECT SUM(c.popn_total)
FROM nyc_neighborhoods AS n
JOIN nyc_census_blocks AS c
  ON ST_Intersects(n.geom, c.geom)
WHERE n.name = 'Battery Park';
```

Übung 4.4 (Bonus)

Welcher Stadtteil hat die höchste Bevölkerungsdichte (Einwohner/km², wobei 1 km² 1000000 m² entspricht)?

Abfrageergebnis

Ergebnis: North Sutton Area (68435 ew_pro_km2), East Village (50404 ew_pro_km2), Chinatown (48825 ew_pro_km2)

Lösung

```
SELECT
  n.name,
  SUM(c.popn_total) / (ST_Area(n.geom) / 1000000) AS ew_pro_km2
FROM nyc_census_blocks AS c
JOIN nyc_neighborhoods AS n
  ON ST_Intersects(c.geom, n.geom)
GROUP BY n.name, n.geom
ORDER BY ew_pro_km2 DESC
LIMIT 3;
```

- Für eine Ausgabe auf zwei Nachkommastellen könnte man ROUND(...::numeric, 2) verwenden; der Cast auf numeric ist nötig, weil PostgreSQL ROUND(float8, integer) nicht kennt.
- Spatial Joins wie hier (... JOIN ... ON ST_Intersects(c.geom, n.geom)) ohne Index können auf grossen Tabellen sehr langsam sein. Ein GiST-Index lässt sich mit dem BefehlCREATE INDEX ON tabelle USING GIST (geom) anlegen. PostGIS prüft dann zuerst mit dem Index, welche Bounding Boxes sich überschneiden, bevor die genaue Geometrieprüfung erfolgt (sog. “Two-Pass”-Verfahren).

Lerneinheit 5 — Koordinatenreferenzsysteme und Projektion

Lernziel: SRID verstehen, Geometrien umprojizieren und die Auswirkungen auf die Messergebnisse beurteilen.

Zeitbedarf: ca. 30 min (Instruktion ~10 min, Übungen ~20 min).

Ein SRID (Spatial Reference ID) funktioniert ähnlich wie die Spracheinstellung einer Karten-App: Dieselbe Stadt kann in verschiedenen Projektionen (Mercator, UTM, WGS 84) dargestellt werden. Ohne zu wissen, in welchem System die Koordinaten vorliegen, ist jede Messung Sinnlos.

Koordinatenreferenzsysteme im Überblick

Ein SRID identifiziert das Koordinatenreferenzsystem (KRS), in dem eine Geometrie vorliegt, eindeutig. Ohne übereinstimmende SRIDs sind räumliche Vergleiche nicht möglich.

Es gibt zwei grundlegende Arten von KRS. Globale KRS wie WGS 84 verwenden geografische Koordinaten (Längen- und Breitengrad in Grad) und sind der Standard für den Datenaustausch, GPS und GeoJSON. Abstands- und Flächenmessungen auf solchen Daten liefern jedoch falsche Ergebnisse, wenn man schlicht `ST_Length` oder `ST_Area` aufruft, da die Koordinateneinheit Grad und nicht Meter ist. Abhilfe schafft entweder eine Umprojektion mit `ST_Transform` oder die Verwendung des PostGIS-Typs `GEOGRAPHY` zusammen mit `ST_Distance` und `ST_Area`, der Abstände direkt auf dem Erdellipsoid berechnet.

Planare KRS hingegen projizieren die Erdoberfläche auf eine Ebene und arbeiten in Metern. Hier gibt es zwei wichtige Untergruppen: Web-Mercator ist die Projektion hinter den meisten Web-Karten. Sie verzerrt Flächen in Polnähe stark und eignet sich daher nicht für Messungen, aber gut für die Darstellung. Lokale planare KRS wie UTM sind dagegen flächentreu und messpräzise, decken aber nur eine begrenzte Region ab.

EPSG-Code	Name	Einheit	Verwendung
4326	WGS 84 (geografisch)	Grad	GPS, GeoJSON, Datenaustausch
3857	Web-Mercator (Pseudo-Mercator)	Meter	Hintergrundkarten
26918	UTM Zone 18N	Meter	NYC-Übungsdatenbank
2831	NY Long Island (State Plane)	Meter	genaue Berechnungen
2056	CH1903+/LV95	Meter	amtliche Landeskarte Schweiz

Wichtige Funktionen im Umgang mit Koordinatenreferenzsystemen:

- `ST_SRID(geom)` — aktuellen SRID abfragen
- `ST_Transform(geom, srid)` — Geometrie in ein anderes KRS umprojizieren.

Werden zwei Geometrien mit unterschiedlichen SRIDs verglichen, liefert PostGIS entweder eine Fehlermeldung oder ein falsches Ergebnis. Prüfen Sie deshalb immer zuerst `ST_SRID(geom)`!

Übungen

Übung 5.1

Welchen SRID haben die Geometrien in `nyc_streets`?

`ST_SRID(geom)` gibt die SRID einer einzelnen Geometrie zurück. Da alle Geometrien einer Tabelle den gleichen SRID haben sollten, reicht `LIMIT 1`.

Abfrageergebnis

Ergebnis: 26918 (UTM Zone 18N)

Lösung

```
SELECT ST_SRID(geom) FROM nyc_streets LIMIT 1;
```

Übung 5.2

Wie gross ist der Unterschied der Gesamtstrassenlänge von NYC, wenn man einmal UTM 18N und einmal in SRID 2831 verwendet?

Zwei separate Abfragen mit `SUM(ST_Length(…))` genügen. `ST_Transform(geom, 2831)` projiziert jede Geometrie on-the-fly um, ohne die Tabelle zu verändern.

Abfrageergebnis

Der Unterschied beträgt ~0.02%. State Plane 2831 ist für NYC optimiert; UTM 18N deckt eine viel grössere Region ab.

Lösung

```
-- Originaldaten in UTM 18N:
SELECT SUM(ST_Length(geom)) AS length_utm18
FROM nyc_streets;

-- Nach Umprojektion in State Plane 2831:
SELECT SUM(ST_Length(ST_Transform(geom, 2831))) AS length_2831
FROM nyc_streets;
```

Übung 5.3

Wo liegt die U-Bahnstation "Broad St" in WGS 84-Koordinaten (Längengrad/Breitengrad)?

Die Funktion `ST_Transform(geom, 4326)` projiziert die UTM-Koordinaten nach WGS 84, die Funktion `ST_AsText(…)` macht das Ergebnis lesbar.

Abfrageergebnis

Ergebnis: ca. -74.01 Länge, 40.71 Breite

Lösung

```
SELECT ST_AsText(ST_Transform(geom, 4326)) AS wgs84
FROM nyc_subway_stations
WHERE name = 'Broad St';
```

Übung 5.4 (Bonus)

Wie viele Strassen kreuzen den 74. Meridian (Längengrad -74)?

Die Geometrie `'SRID=4326;LINESTRING(-74 20, -74 60)'` ist eine vertikale Linie entlang des Meridians.

Abfrageergebnis

Ergebnis: 223 Strassen

Lösung

```
SELECT Count(*)
FROM nyc_streets
WHERE ST_Intersects(
  ST_Transform(geom, 4326),
  'SRID=4326;LINESTRING(-74 20, -74 60)'
);
```

Für Distanzmessungen über grosse Entfernungen (z.B. Zürich nach New York) sollte man den Typ `GEOGRAPHY` statt `GEOMETRY` verwenden, da er auf dem Erdellipsoid rechnet und somit korrekte Abstände in Metern liefert, ohne dass eine passende Projektion gewählt werden muss.

Abschluss

Die Lerneinheiten decken die wichtigsten Konzepte von Spatial SQL bzw. PostGIS ab: vom Erkunden eines räumlichen Datenbankschemas über Messfunktionen und räumliche Prädikate bis hin zu Spatial Joins und Koordinatenreferenzsystemen.

In diesem Tutorial werden bewusst nur die Grundlagen behandelt. Folgende Themen bauen darauf

auf:

Geography-Typ
: Abstände auf der Erdkugel berechnen ohne Projektion (z.B. Zürich nach New York)

Spatial Indexing (GIST)
: räumliche Abfragen auf grossen Datensätzen stark beschleunigen

`ST_Buffer` / `ST_Union` / `ST_Intersection`
: Geometrien konstruieren und kombinieren

`ST_Centroid` / `ST_PointOnSurface`
: repräsentative Punkte für Flächen berechnen

Nearest-Neighbour (`<->`)
: den nächstgelegenen Punkt effizient finden

QGIS
: Abfrageergebnisse aus PostGIS direkt auf einer Karte visualisieren

Der NYC-Datensatz eignet sich ideal zum Lernen. Für eigene Übungen und Projekte bietet sich die freie Quelle OpenStreetMap mit weltweiten Vektordaten an. Auf OpenSchoolMaps gibt es dazu das Tutorial "OpenStreetMap-Daten extrahieren mit SQL und Postpass".

Dies sind weitere Tutorials und Ressourcen:

- Introduction to PostGIS Workshop (EN) — der vollständige Originalworkshop mit allen hier weggelassenen Themen
- PostGIS Dokumentation — vollständige Funktionsreferenz aller `ST_`-Funktionen
- epsg.io — Nachschlagewerk für alle Koordinatenreferenzsysteme

---

APPENDIX: Schema der NYC-Übungsdatenbank

Anbei finden Sie einen vollständigen Auszug aus dem Schema der NYC-Übungsdatenbank.

Alle Tabellen beginnen mit `nyc_` und alle Geometrien liegen im Koordinatensystem UTM Zone 18N (EPSG:26918).

Die Tabelle `nyc_census_blocks` — Volkszählungsblöcke (38794 Datensätze): Ein Census Block ist die kleinste Einheit, für die Volkszählungsdaten erfasst werden. Aus ihrer Vereinigung entstehen alle grösseren Gebietseinheiten.

`blkid`
: 15-stelliger eindeutiger Schlüssel, z. B. 360050001009000

`popn_total`

Gesamtbevölkerung des Blocks

`popn_white`

Bevölkerung, die sich als "White" identifiziert

`popn_black`

Bevölkerung, die sich als "Black" identifiziert

`popn_nativ`

Bevölkerung, die sich als "Native American" identifiziert

`popn_asian`

Bevölkerung, die sich als "Asian" identifiziert

`popn_other`

Bevölkerung mit anderen Angaben

`boroname`

Name des Boroughs: Manhattan, The Bronx, Brooklyn, Staten Island, Queens

`geom`

Polygongeometrie des Blocks

Tabelle `nyc_neighborhoods` — Stadtteile (129 Datensätze): Stadtteile sind soziale Konstrukte ohne feste amtliche Grenzen — sie spiegeln historische und kulturelle Wahrnehmungen wider, nicht Verwaltungsgrenzen.

`name`

Name des Stadtteils, z.B. "West Village"

`boroname`

Name des Boroughs

`geom`

Polygongeometrie des Stadtteils

Tabelle `nyc_streets` — Strassennetz (19091 Datensätze). Das Strassenmittelliniennetz der Stadt, mit Klassifizierungen nach Strassentyp.

`name`

Strassenname

`oneway`

Einbahnstrasse? "yes" = ja, "" = nein

`type`

Strassentyp: primary, secondary, residential, motorway, ...

`geom`

Liniengeometrie (MultiLineString)

Tabelle `nyc_subway_stations` — U-Bahn-Stationen (491 Datensätze)

`name`

Stationsname

`borough`

Borough der Station

`routes`

Linien, die durch die Station fahren, z.B. "A,C,E"

`transfers`

Linien, auf die man umsteigen kann

`express`

Haelt ein Expresszug? "express" = ja, "" = nein

`geom`

Punktgeometrie der Station

Tabelle `nyc_census_sociodata` — Sozioökonomische Daten (keine Geometrie): Daten auf Ebene der Census Tracts (grösser als Census Blocks). Verknüpfung mit den räumlichen Tabellen über `tractid`.

`tractid`

Fremdschlüssel zu den Census Tracts

`transit_*`

Pendlerverhalten (Verkehrsmittelwahl, Pendelzeit)

`family_income_*`

Haushaltseinkommen nach Einkommensklassen

`edu_*`

Bildungsabschlüsse der Bevölkerung

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