Achtung

Die Beispiele in diesem Vortrag beziehen sich auf die alte Version 0.1 der Postpass-API.

Aktuell ist Version 0.20. Die Tabellen heißen jetzt anders, allerdings gibt es „Kompatibilitäts-Views“, so dass viele Beispiele noch funktionieren werden, wenn man im URL die 0.1 durch die 0.2 austauscht.

Eine aktuelle Version der meisten Beispiele ist auf github.com/woodpeck/postpass-ops/examples.

Beispiel für den Fast-Food-Query nach neuem API:

```
curl -g http://postpass.geofabrik.de/api/0.2/interpreter \
   --data-urlencode "data=
     SELECT tags, geom
     FROM postpass_point
     WHERE tags->>'amenity'='fast_food'
     AND geom && st_setsrid(st_makebox2d(
        st_makepoint(8.34,48.97),
        st_makepoint(8.46,49.03)), 4326)"
```

OVERPASS TURBO GOES POSTGIS

Frederik Ramm <frederik@remote.org>

Münster, 27.3.2025

OPENSTREETMAP-DATENBANKEN

- Standard-API ("editing API")
- 2007: "XAPI" (2011: jXAPI)
- 2011: Overpass API
- 2013: Overpass Turbo

Overpass API Query Form

Query

extract POIs for a region

1

I am currently working on a list of available hotels in the Republic of Georgia. As the OSM-Database has lots of POIs, is it possible to access this data for that purpose? I would like to run a query to retrieve hotels with addresses, phone numbers and other details that are entered. The coordinates would not matter for my purpose.

The result should be an Excel-file or similar table format.

How can I do this? All help appreciated!

query pois lists database

asked **08 May '12, 12:26**

moszkva ter
2.1k●22●39●60
accept rate: 17%

Follow this question

By Email:
Once you sign in you will be able to subscribe for any updates here

By RSS:
Answers

Answers and Comments

This was the support site for OpenStreetMap. It has now been replaced by community.osm.org.

Question tags:

4

You can get the data in JSON format in two steps from Overpass API:

First, obtain the area by pasting the following request

```
<osm-script output="json">

<coord-query lat="42.2" lon="43.3"/>
<print/>

</osm-script>
```

into the Overpass API query form.

From the result, you can read off the area with name "Georgia" that the relevant area id is 3600028699.

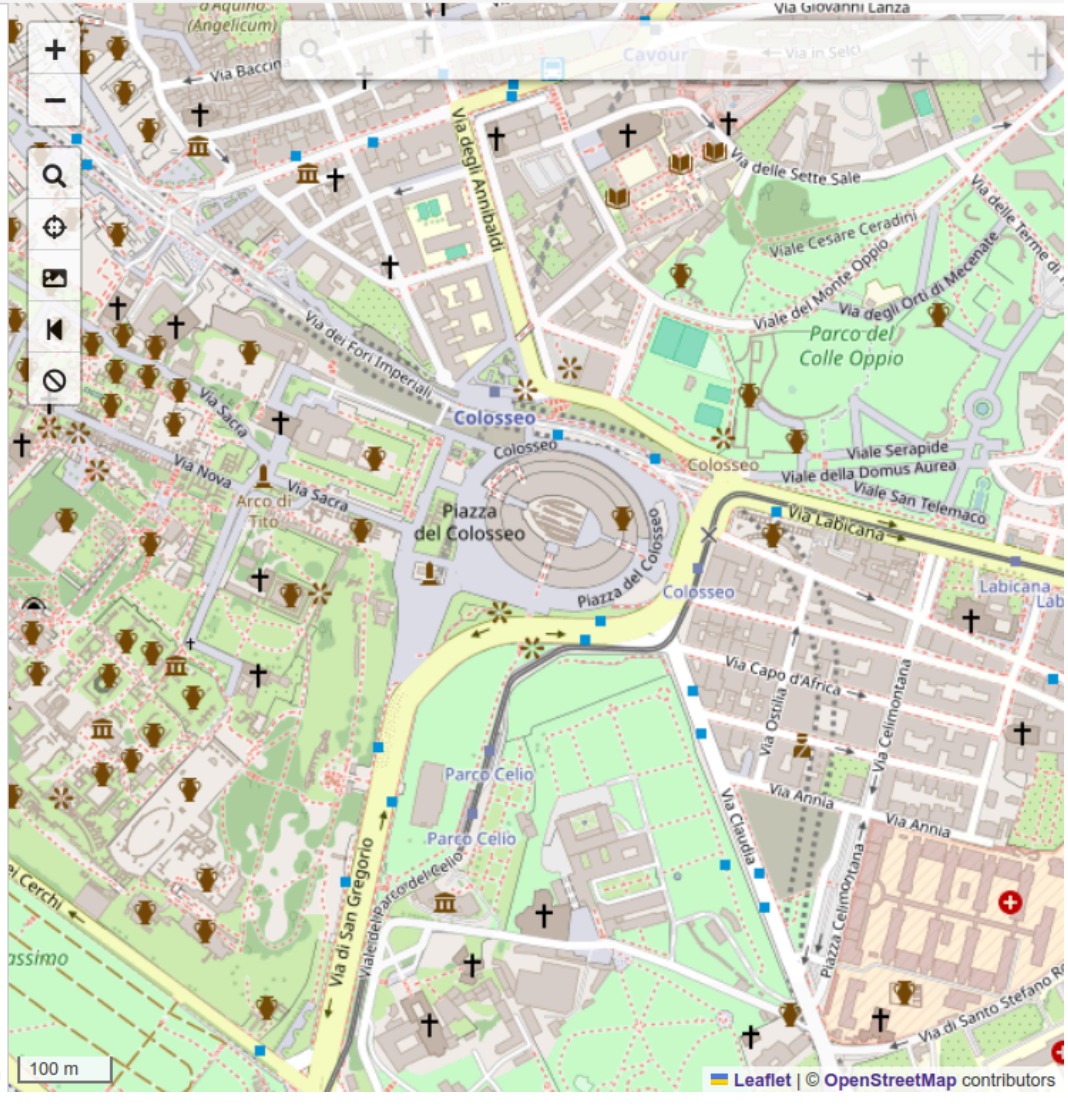
Now you can query for the really wanted data, again on Overpass API query form:

```
<osm-script output="json">

<query type="node">
  <area-query ref="3600028699"/>
  <has-kv k="tourism" v="hotel"/>
</query>
<print/>
<area-query ref="3600028699"/>
<query type="way">
  <recurse type="node-way"/>
  <has-kv k="tourism" v="hotel"/>
</query>
<print/>

</osm-script>
```

This results in a 92 KB big JSON file containing all information about nodes or ways in Georgia tagged with `tourism=hotel`. If you want additionally hostels, repeat the second step with `tourism=hostel` and so on.

Run | Share | Export | Wizard | Save | Load | Settings | Help | **overpass turbo** | Map | Data

```
/*
This is an example Overpass query.
Try it out by pressing the Run button above!
You can find more examples with the Load tool.
*/
node
  [amenity=drinking_water]
  ({{bbox}});
out;
```



Download amenities kmz file

0

Hello, I would like to download amenities of a certain type and area, preferably in kmz format. For example I would like to have a kmz file of amenity=drinking_water in Berlin (more or less). How to do it?

download export amenities kmz

asked **24 Aug '21, 12:35**

Leszek
11●1●1●2
accept rate: 0%

One Answer:

active answers | oldest answers | newest answers | popular answers

1

You can use overpass turbo to retrieve specific POIs from OSM.

The following query will search for all drinking water POIs in the currently visible area:

```
[out:json][timeout:25];
// gather results
(
  nwr["amenity"="drinking_water"]({{bbox}});
  nwr["drinking_water"="yes"]({{bbox}});
);
// print results
out body;
>;
out skel qt;
```

You can run it here. Hit run. Then go to Export -> Save as KML.

permanent link

answered **28 Oct '21, 10:11**

scai ♦

33.3k●21●309●459

accept rate: 23%

Thank you very much!

Leszek (28 Oct '21, 10:16)

download ×275

kmz ×7

question asked: **24 Aug '21, 12:35**

question was seen: **1,933 times**

last updated: **28 Oct '21, 10:16**

Related questions

Is it possible to export an OSM Cycle map in order to work on it in Adobe Illustrator

Exporting maps

Full zoomed map download,

how can I download a map in vectors???

Download OSM data - button disappeared

where to get a lat long database for India with

Gather amenities sorted by county borders

0

Hello everyone,

my goal is to gather the number of certain amenitys (for example: amenity=bar) in Germany, but sorted in a csv by the county border (admin_level=6) they belong to.

This works fine for one county border in Overpass Turbo:

```
[out:csv(::type,::id, name,"addr:postcode", "addr:city")];
area[admin_level=6]["name"="Bautzen"];
(
   node["amenity"="bar"](area);
   way["amenity"="bar"](area);
   rel["amenity"="bar"](area);
);
out body;
>;
out skel qt;
```

But Germany has 403 of them, so inserting each of them in a new query would be way to much effort. On the other hand I don't now how to match the amenitys, when I do a seach for the whole of Germany. For example, this isn't possible via the postcode, because not every amenity have one assigned.

Question tags:

overpass-turbo ×228

amenity ×94

csv ×42

region ×17

One Answer:

active answers | oldest answers | newest answers | popular answers

3

Since it looks like you want to make many of these queries, don't hammer someone else's server with it - do it yourself. Load the OSM data for Germany into a PostGIS database with osm2pgsql, then run this query:

```
SELECT
  county.name, count(*)
FROM
  planet
```

[redacted]

```
SELECT
  poi.amenity, county.name, count(*)
FROM
  planet_osm_point poi,
  planet_osm_polygon county
WHERE
  poi.amenity in ('bar', 'restaurant', 'pub')
  AND county.boundary='administrative'
  AND county.admin_level='6'
  AND st_contains(county.way, poi.way)
GROUP BY poi.amenity, county.name;
```

(BTW, note that these queries will omit Hamburg and Berlin due to lack of an adminlevel 6 boundary.)

If you are interested in OSM statistics then using your own PostGIS and learning a little SQL is really useful.

permanent link

answered **26 Nov '19, 17:38**

Frederik Ramm ♦

82.5k●92●720●1273

accept rate: 23%

something specific

overpass // csv output // how can i get the region of nodes out

Overpass turbo query - OSM Cities in CSV with tags

CSV of nodes with city name located in country

How to add factories

How can I add

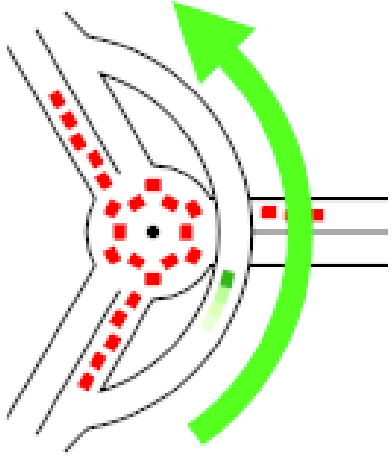
Searching for car_sharing

PROBLEM:

- Keiner hat "mal eben" eine PostGIS mit OSM-Daten zur Hand
- viele haben nicht mal Linux am Start
- keiner ist so doof, eine weltweit offene PostGIS zu betreiben

ODER?

INTRODUCING POSTPASS



Overpass
API

PostGIS

INTRODUCING POSTPASS

- Serversoftware ist ein kleiner Wrapper um PostGIS: github.com/woodpeck/postpass
- Betrieb derzeit auf Geofabrik-Server: github.com/woodpeck/postpass-ops

SO GEHT'S:

```
curl -g http://postpass.geofabrik.de/api/0.1/interpreter \
   --data-urlencode "data=
     SELECT name, way
     FROM planet_osm_point
     WHERE amenity='fast_food'
       AND way && st_setsrid(st_makebox2d(
          st_makepoint(8.34,48.97),
          st_makepoint(8.46,49.03)), 4326)"
```

→ Rückgabe ist eine GeoJSON-FeatureCollection

```
curl -g http://postpass.geofabrik.de/api/0.1/interpreter \
   --data-urlencode "data=
     DROP TABLE planet_osm_point"
```

- Backend parst das SQL nicht selbst
- verwendet aber einen readonly-Useraccount

ÜBERLASTSCHUTZ

Was tun mit Queries wie "select * from planet_osm_point"?

- Queries werden zunächst in ein EXPLAIN() gepackt (Nebeneffekt: Syntax-Check)
- Je nach von PostgreSQL geschätzter Komplexität kommen sie in eine von drei Warteschlangen
- Warteschlangen für "schnelle" Requests haben mehr worker

BEISPIELE

FAST FOOD IN KARLSRUHE

Overpass

```
[out:json];
node["amenity"="fast_food"]
   (48.97,8.34,49.03,8.46);
out;
```

ca. 400ms

Postpass

```
SELECT name,way,tags
FROM planet_osm_point
WHERE amenity='fast_food'
   AND way && st_setsrid(st_makebox2d(
      st_makepoint(8.34,48.97),
      st_makepoint(8.46,49.03)), 4326)
```

ca. 200ms

FAST FOOD IN KARLSRUHE

Overpass

```
data=[out:json];
nwr["amenity"="fast_food"]
   (48.97,8.34,49.03,8.46);
out geom;
```

ca. 500ms

Postpass

```
SELECT name,way,tags
FROM planet_osm_point
WHERE amenity='fast_food'
   AND way && st_setsrid(st_makebox2d(
      st_makepoint(8.34,48.97),
      st_makepoint(8.46,49.03)), 4326)
UNION
SELECT name,way,tags
FROM planet_osm_polygon
WHERE amenity='fast_food'
   AND way && st_setsrid(st_makebox2d(
      st_makepoint(8.34,48.97),
      st_makepoint(8.46,49.03)), 4326)
```

ca. 300ms

STRASSENBAHNLINIE 3 IN KARLSRUHE

Overpass

```
rel[ref=3][route=tram]
   (48.97,8.34,49.03,8.46);>;
out;
```

ca. 600ms

ca. 730ms für alle

Postpass

```
SELECT name,way,tags
FROM planet_osm_line
WHERE route='tram'
   AND ref='3'
   AND way && st_setsrid(st_makebox2d(
      st_makepoint(8.34,48.97),
      st_makepoint(8.46,49.03)), 4326)
```

ca. 150ms

ca. 240ms für alle

ADDR:STREET OHNE ADDR:POSTCODE

Overpass

```
[out:json]
  nw["addr:street"]
     [!"addr:postcode"]
        (48.97,8.34,49.03,8.46);
>;
out skel;
```

ca. 700ms

Postpass

```
SELECT osm_id,way
FROM planet_osm_point
WHERE tags?'addr:street'
   AND NOT tags?'addr:postcode'
   AND way && st_setsrid(st_makebox2d(
      st_makepoint(8.34,48.97),
      st_makepoint(8.46,49.03)), 4326)
UNION
SELECT osm_id,way
FROM planet_osm_polygon
WHERE tags?'addr:street'
   AND NOT tags?'addr:postcode'
   AND way && st_setsrid(st_makebox2d(
      st_makepoint(8.34,48.97),
      st_makepoint(8.46,49.03)), 4326)
```

ca. 240ms

DINGE MIT REF=15398

Overpass

```
[out:json]
nwr[ref="15398"]
>;
out;
```

ca. 1,2s

Postpass

```
SELECT osm_id, way
FROM planet_osm_point
WHERE ref='15398'
UNION
...
```

ca. 50s (!)

VERWALTUNGSGRENZEN, DIE "MÜNCHEN" HEISSEN

Overpass

```
[out:json]
r[name="München"]
 [boundary="administrative"]
>;
out;
```

ca. 0,4s

Postpass

```
SELECT osm_id, way
FROM planet_osm_polygon
WHERE name='München'
AND boundary='administrative'
```

ca. 0,4s

(hierfür existiert ein Index)

UNTER DER HAUBE

- "Standard"-OpenStreetMap-Carto-Import...
- ... aber mit EPSG:4326 (osm2pgsql-Flag -l)
- einige Extra-Indexe
- aus jeder Zeile wird per ST_AsGeoJSON ein Feature
- die Zeilen werden mit json_agg zu einer FeatureCollection zusammengefasst

FLAGS

- options[pretty]=false – kein Pretty Print
- options[geojson]=false – kein GeoJSON
- options[collection]=false – überhaupt kein json_agg (aber: Query muss einen Skalar zurückgeben)

QUERY OHNE GEOMETRIE

Wie viele von welchen amenity-Typen gibt es in der Boundingbox?

```
curl -g https://postpass.geofabrik.de/api/0.1/interpreter \
   --data-urlencode "options[geojson]=false"
   --data-urlencode "data=
    SELECT count(*),amenity
    FROM planet_osm_point
        WHERE amenity is not null
        AND way && st_setsrid(st_makebox2d(
           st_makepoint(8.34,48.97),
           st_makepoint(8.46,49.03)), 4326)
    GROUP BY amenity"
```

(0,4s)

FALSCHE POSTLEITZAHLEN IN DEUTSCHLAND

```
curl -g https://postpass.geofabrik.de/api/0.1/interpreter \
   --data-urlencode "data=
   SELECT
      p.osm_id, p.way, p.tags->'addr:postcode' as punkt_plz,
      plz.tags->'postal_code' as poly_plz
   FROM planet_osm_point p, planet_osm_polygon plz
   WHERE plz.way && st_setsrid(st_makebox2d(
      st_makepoint(5.53,47.23), st_makepoint(15.38,54.96)), 4326)
   AND st_contains(plz.way,p.way)
   AND p.tags->'addr:postcode' <> plz.tags->'postal_code'
   AND plz.boundary='postal_code'"
```

(50s)

SCHNITTFLÄCHEN VON LANDNUTZUNGEN

```
curl -g https://postpass.geofabrik.de/api/0.1/interpreter \
   --data-urlencode "data=
   SELECT
      st_intersection(a.way, b.way) as inter,
      a.landuse as landuse1, b.landuse as landuse2
   FROM planet_osm_polygon a, planet_osm_polygon b
   WHERE a.osm_id < b.osm_id
   AND a.way && st_setsrid(...)
   AND b.way && st_setsrid(...)
   AND a.landuse is not null
   AND b.landuse is not null
   AND st_overlaps(a.way, b.way)"
```

INTEGRATION MIT OVERPASS TURBO

- Overpass Turbo kannte schon immer einen Selektor für Datenquellen: {{data:...}}
- Overpass Turbo wandelt die von Overpass gelieferten Daten in GeoJSON um – bei Postpass nicht notwendig
- nur minimale Änderungen waren notwendig, um Postpass zu unterstützen
- seit gestern live (danke Martin!)

INTEGRATION MIT OVERPASS TURBO

```
{{data:sql,server=https://postpass.geofabrik.de/api/0.1/}}

SELECT name,way,tags
FROM planet_osm_point
WHERE amenity='fast_food'
   AND way && {{bbox}}
```

BEIM BASTELN BEACHTEN

- einige Spalten sind "echte" Datenbankspalten, andere im "tags"-hstore (Zugriff über tags->'key')
- immer eine Geometriespalte im SELECT
- brauche ich planet_osm_point, line, polygon?
- ... und kein Semikolon am Ende

ROOM FOR IMPROVEMENT

- gerne den Source auf Github anschauen und verbessern!
- Fehlermeldungen könnten besser sein
- vielleicht kann man die json_agg-Magie transparenter machen
- Instrumentierung für Performance-/Last-Beobachtung
- Query-Timeouts
- ...

DANKE

Frederik Ramm <frederik@remote.org>

(Session am OSM-Samstag?)