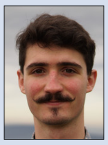
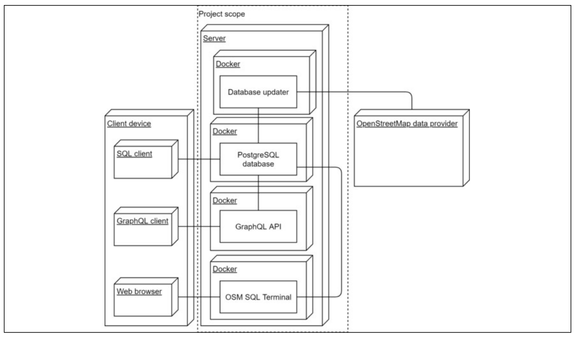

Philipp
Bolliger

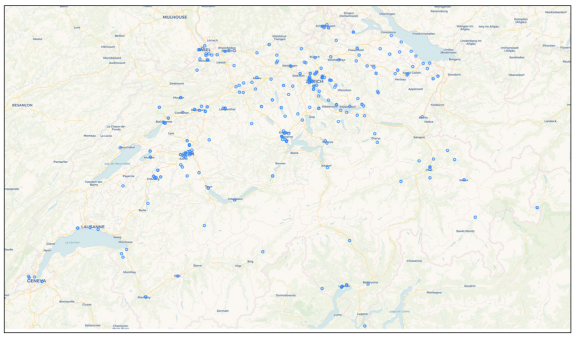
Marc
Scherrer

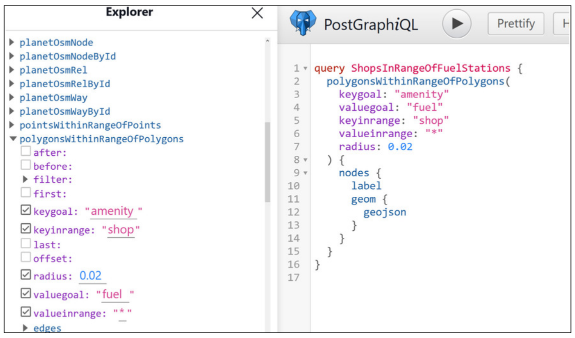
# An Analysis Platform for OpenStreetMap with Spatial SQL and GraphQL

## Extended OpenStreetMap Database Two (EOSMDBTwo)



Deployment diagram of EOSMDBTwo
Own presentment


Map showing all shops within 20 meters of a fuel station as result of a GraphQL or SQL query
Own presentment



```
planetOsmNode
planetOsmNodeById
planetOsmRel
planetOsmRelById
planetOsmWay
planetOsmWayById
pointsWithinRangeOfPoints
polygonsWithinRangeOfPolygons
  after:
  before:
  filter:
  first:
  keygoal: "amenity "
  keyinrange: "shop"
  last:
  offset:
  radius: 0.02
  valuegoal: "fuel "
  valueinrange: "*"
  edges
```

```
query ShopsInRangeOfFuelStations {
  polygonsWithinRangeOfPolygons(
    keygoal: "amenity"
    valuegoal: "fuel"
    keyinrange: "shop"
    valueinrange: "*"
    radius: 0.02
  ) {
    nodes {
      label
      geom {
        geojson
      }
    }
  }
}
```

PostGraphiQL: Interface as implemented using PostGraphile showing the GraphQL query which produced figure 2.
Own presentment

Introduction: OpenStreetMap (OSM) provides geospatial data of the world which contains massive information that offers an almost untapped potential for spatial data analytics. Yet the efficient processing of this kind of data remains a big data challenge.

One of the existing approaches is the EOSMDBOne (Extended OpenStreetMap Database One), which covers just Switzerland and Liechtenstein, based on a PostgreSQL database and the PostGIS extension, which offers geospatial calculation functions. However, the current database schema is sub-optimal for certain geospatial queries, and some of the data pre-processing tools involved are no longer maintained.

Definition of Task: The main task of this work was to implement the EOSMDBTwo with an adapted database schema as compared to EOSMDBOne. And it must be based on up-to-date OSM data pre-processing tools. Other tasks include allowing for more flexible configurations e.g. of data sources as OSM data extractions, and maintaining responsiveness of the PostgreSQL database.

SQL was given being the natural API to databases. In addition - since GraphQL became an important alternative to REST services - another requirement was to evaluate and implement a GraphQL API service. Furthermore, a SQL web frontend should be created, called OSM SQL Terminal, where data from the EOSMDBTwo can be queried and visualized as an interactive web map.

Result: The software tool chain to load OSM data from well-known data repositories was evaluated. The resulting tools used are PyOsmium and osm2pgsql. The database schema was revised based on experiences of using the EOSMDBOne.

The EOSMDBTwo contains functionality for configuring a set of OSM extracts to be initially imported. The EOSMDBTwo then updates the data from the extract automatically in a defined time interval. To improve query performance, indices were defined and the PostgreSQL database was optimized for OSM data and this specific geospatial data analysis use case.

Additionally, a GraphQL API was created, providing functionality to query the EOSMDBTwo as an alternative to (spatial) SQL. Since GraphQL is based on JSON, the output of data types geometry and geography are in GeoJSON. A custom function was added, which executes a more complex query, that searches all objects that are in a certain range of other objects. The OSM SQL Terminal continued as separate project outside of the scope of this work.

The documentation of EOSMDBTwo contains educational show cases, and in conjunction with the OSM SQL Terminal it is ready to replace EOSMDBOne. It is easily deployable thanks to containerization with Docker.