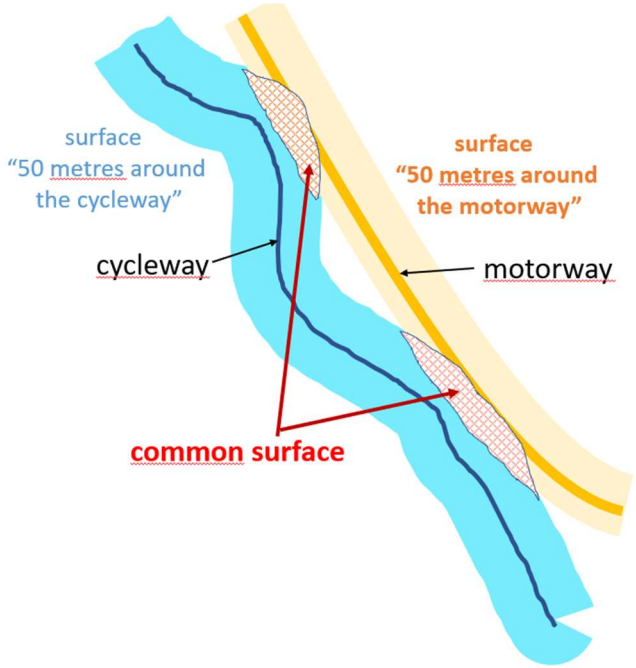
Calculation of new tags :



As example for the noise calculation, we have 2 OSM-objects :

a- an highway (=cycleway or track…), defined as « line » in OSM

b- an highway (=motorway or primary..), defined as « line » in OSM

If we now consider the « surfaces » along these lines (defined with a distance to the line < 50 m), the « intersection » of the 2 surfaces is the key for calculation:

(it integrates the distance of the 2 objects along the segment considered)

Next, using spatial SQL, we can « easily » calculate a value for the « noise » tag above :

(assuming « cyclew » is the object of the cycleway, « motorw » is the object of the motorway)

⇨ st_area(st_intersection(ST_Buffer(cyclew.way, 30), ST_Buffer(motorw.way, 50))) / st_area(ST_Buffer(cyclew .way, 50)

The result is a factor (0 <= value <= 1) indicating the noice level!!!!!!

Next challenge: how to find objects within a distance?!

As example for Germany only, nearly 18,000,000 of “lines” and 43,000,000 of “polygons” exist.

Again, the “spatial database” offers the solution:

Example: (“JOIN” part of a sql)

```
FROM planet_osm_line AS m
INNER JOIN planet_osm_polygon AS q ON ST_DWithin(m.way, q.way, 120)
WHERE  m.highway is not null
and
 q.natural in ('water') and (q.water is null or q.water not in ('wastewater'))
```

This giving all combinations of “highway” and “polygon” of type “water” having a distance within 120 m.

As the function “ST_DWithin” is supported by the spatial indexes, the sql remains performant in the big database.

More information :

(About the database)

https://osm2pgsql.org/doc/manual.html

https://www.postgresql.org/docs/14/tutorial-createdb.html

(A first test started in 2020)

https://trailrouter.com/blog/how-trail-router-works

SQL Examples:

Noise factor SQL

```
SELECT     losmid, lhighway, sum(noise_factor) sum_noise_factor
into table noise_tmp
from (
SELECT
    m.osm_id losmid, m.highway lhighway,
    case
    when q.highway in ('motorway', 'motorway_link','trunk','trunk_link') then
         sum(st_area(st_intersection(ST_Buffer(m.way, 50),
ST_Buffer(q.way, 50)))
          / st_area(ST_Buffer(m.way, 50)))
    when q.highway in ('primary','primary_link') then
      sum(st_area(st_intersection(ST_Buffer(m.way, 50), ST_Buffer(q.way,
50)))
          / (2 * st_area(ST_Buffer(m.way, 50))))
    when q.highway in ('secondary') then
    sum(st_area(st_intersection(ST_Buffer(m.way, 50), ST_Buffer(q.way,
50)))
          / (6 * st_area(ST_Buffer(m.way, 50))))
    end
as noise_factor
FROM planet_osm_line AS m
INNER JOIN planet_osm_line AS q ON ST_DWithin(m.way, q.way, 100)
WHERE m.highway is not null
and q.highway in ('motorway',
'motorway_link','trunk','trunk_link','primary','primary_link','secondary')
GROUP BY losmid, lhighway, q.highway
order by noise_factor desc)
as abcd
GROUP BY losmid, lhighway
order by sum_noise_factor desc;
```

A further sql is used to define “classes” as used in the tags.

```
SELECT    losmid,
case
  when y.sum_noise_factor < 0.1  then null
```

```
  when y.sum_noise_factor < 0.17  then '1'
  when y.sum_noise_factor < 0.27  then '2'
  when y.sum_noise_factor < 0.4    then '3'
  when y.sum_noise_factor < 0.75  then '4'
  when y.sum_noise_factor < 1.5    then '5'
  else '6'
end as noise_class
into table noise_tags
from noise_tmp y
where y.sum_noise_factor > 0.1;
```

Note : All highway types should be calculated and loaded in rd5 (to be able to apply the same noise penalty for all highways), else the routing « could » as example route on a primary instead of the cycleway on the side!

RIVER / SEE SQL

```
select xid , sum(water_river_see) as river_see
into table river_tmp
from (
SELECT     m.osm_id  as xid,
   sum(
      st_area(st_intersection(ST_Buffer(m.way, 150), ST_Buffer(q.way,
150)))      /
      st_area(ST_Buffer(m.way, 150))
   ) as water_river_see
FROM planet_osm_line AS m
INNER JOIN planet_osm_polygon AS q ON ST_DWithin(m.way, q.way,
120)
WHERE  m.highway is not null
and
 q.natural in ('water') and (q.water is null or q.water not in ('wastewater'))
GROUP BY m.osm_id
union
SELECT    m.osm_id as xid,
   sum(
      st_area(st_intersection(ST_Buffer(m.way, 150), ST_Buffer(q.way,
150)))     /
      st_area(ST_Buffer(m.way, 150))
   ) as water_river_see
FROM planet_osm_line AS m
```

```
INNER JOIN planet_osm_line AS q ON ST_DWithin(m.way, q.way, 120)
WHERE m.highway is not null
and
q.waterway in ('river','canal')
GROUP BY m.osm_id
) as abcd
GROUP BY xid
order by river_see desc;


SELECT     y.xid losmid,
case
  when y.river_see < 0.1  then null
  when y.river_see < 0.35 then '1'
  when y.river_see < 0.55 then '2'
  when y.river_see < 0.75 then '3'
  when y.river_see < 1.3  then '4'
  when y.river_see < 1.8  then '5'
  else '6'
end as river_class, ' ende'
into table river_tags
from river_tmp y where y.river_see > 0.1;
```

Forest / park

```
SELECT
   l.osm_id, l.highway,
   sum(
      st_area(st_intersection(ST_Buffer(l.way, 50), ST_Buffer(p.way,
50)))
      /
      st_area(ST_Buffer(l.way, 50))
   ) as green_factor
into table forest_tmp
FROM planet_osm_line AS l
INNER JOIN planet_osm_polygon AS p ON ST_DWithin(l.way, p.way,
70)
WHERE l.highway is not null
and
```

```
(p.landuse in
('forest','allotments','flowerbed','orchard','vineyard','recreation_ground','vill
age_green')
    or p.leisure in ( 'park','nature_reserve'))
GROUP BY l.osm_id, l.highway
order by green_factor desc;

SELECT  y.osm_id losmid,
case
  when y.green_factor < 0.1   then null
  when y.green_factor < 0.33  then '1'
  when y.green_factor < 0.65  then '2'
  when y.green_factor < 0.95  then '3'
  when y.green_factor < 1     then '4'
  when y.green_factor < 1.2   then '5'
  else '6'
end as forest_class, ' ende'
into table forest_tags
from forest_tmp y where y.green_factor  > 0.1;
```