

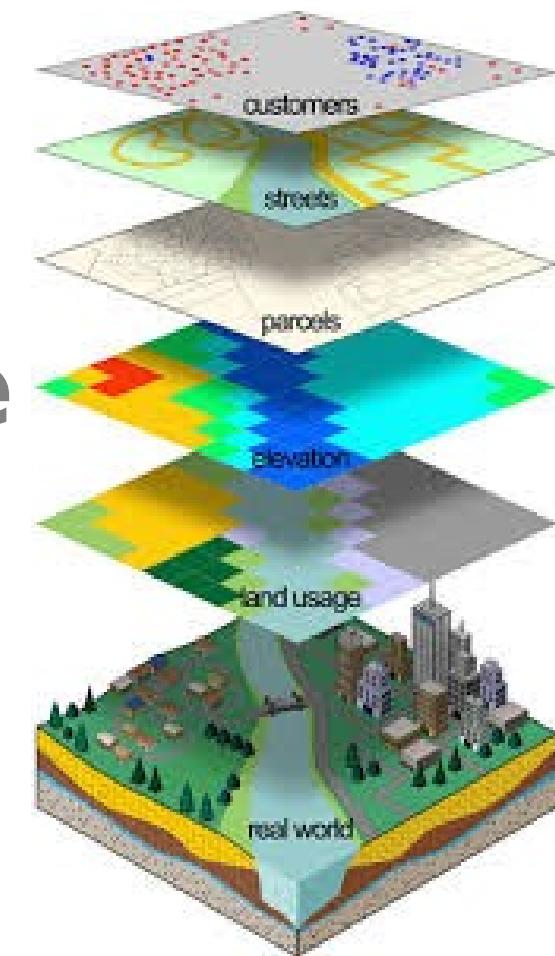
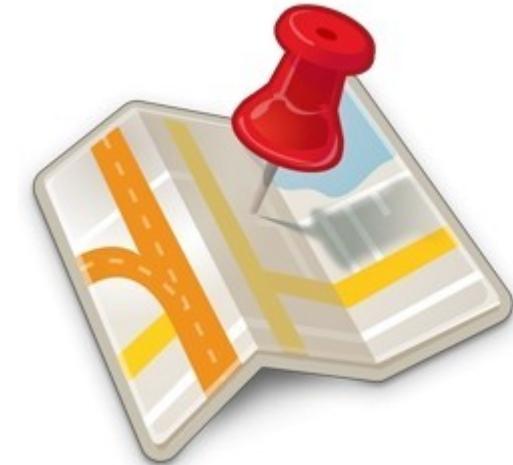
# Beyond PostGIS basics



## More spatial !

# **P**ost - **G**eographical **I**nformation **S**ystem

**Capture, create, store,  
analyze, share, visualize  
data related to space**



# PostGIS



# PostGIS 101

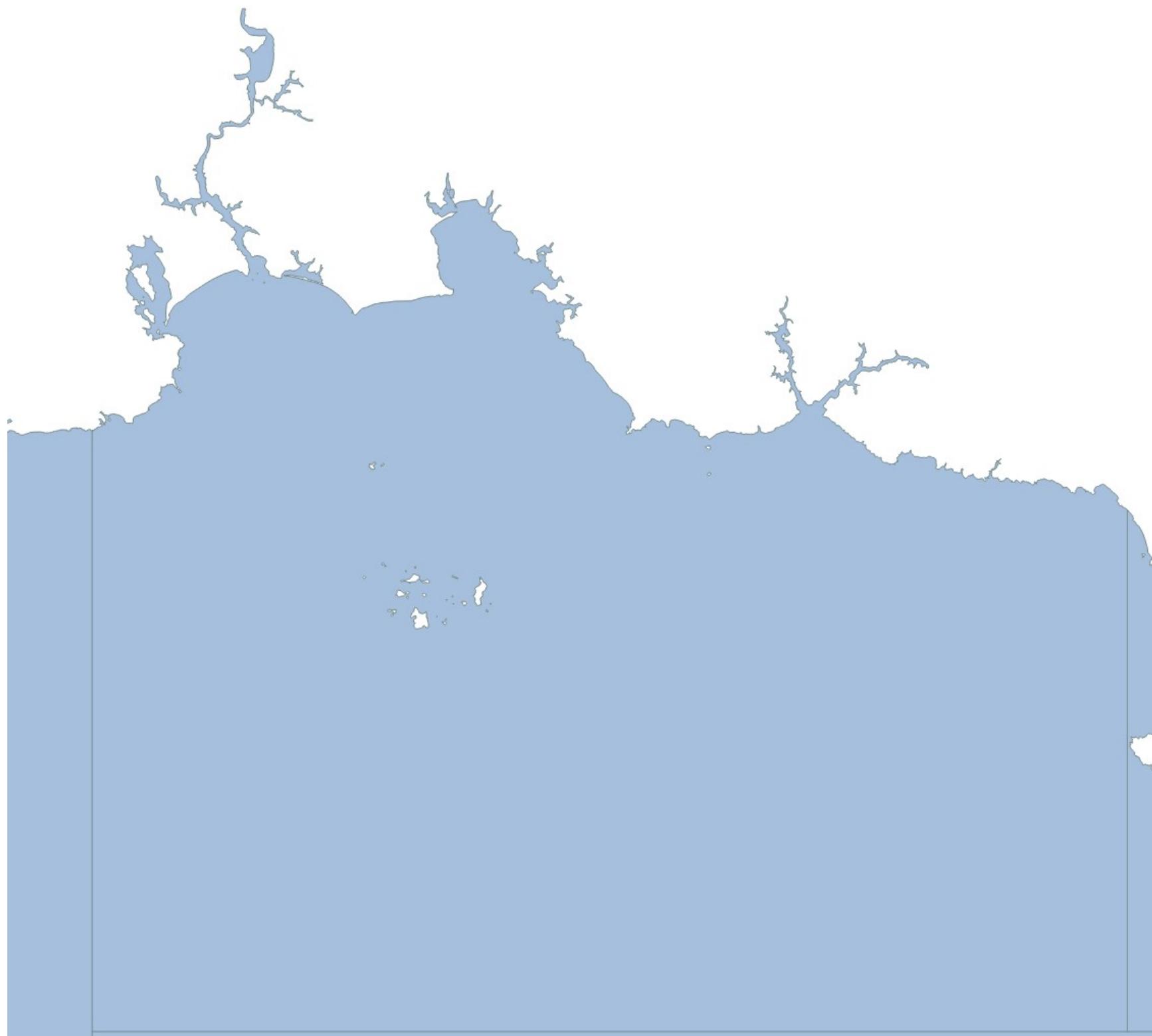
- › PostgreSQL Extension
- › Data types : point, lines, polygons...
- › Spatial indexes
- › Lots of functions

```
SELECT
    d.* , count(*) / ST_Area(d.geom)
FROM
    clients as c
JOIN
    districts as d
ON
    ST_Contains(d.geom, c.geom)
GROUP by
    d.gid;
```

# PostGIS 2.2

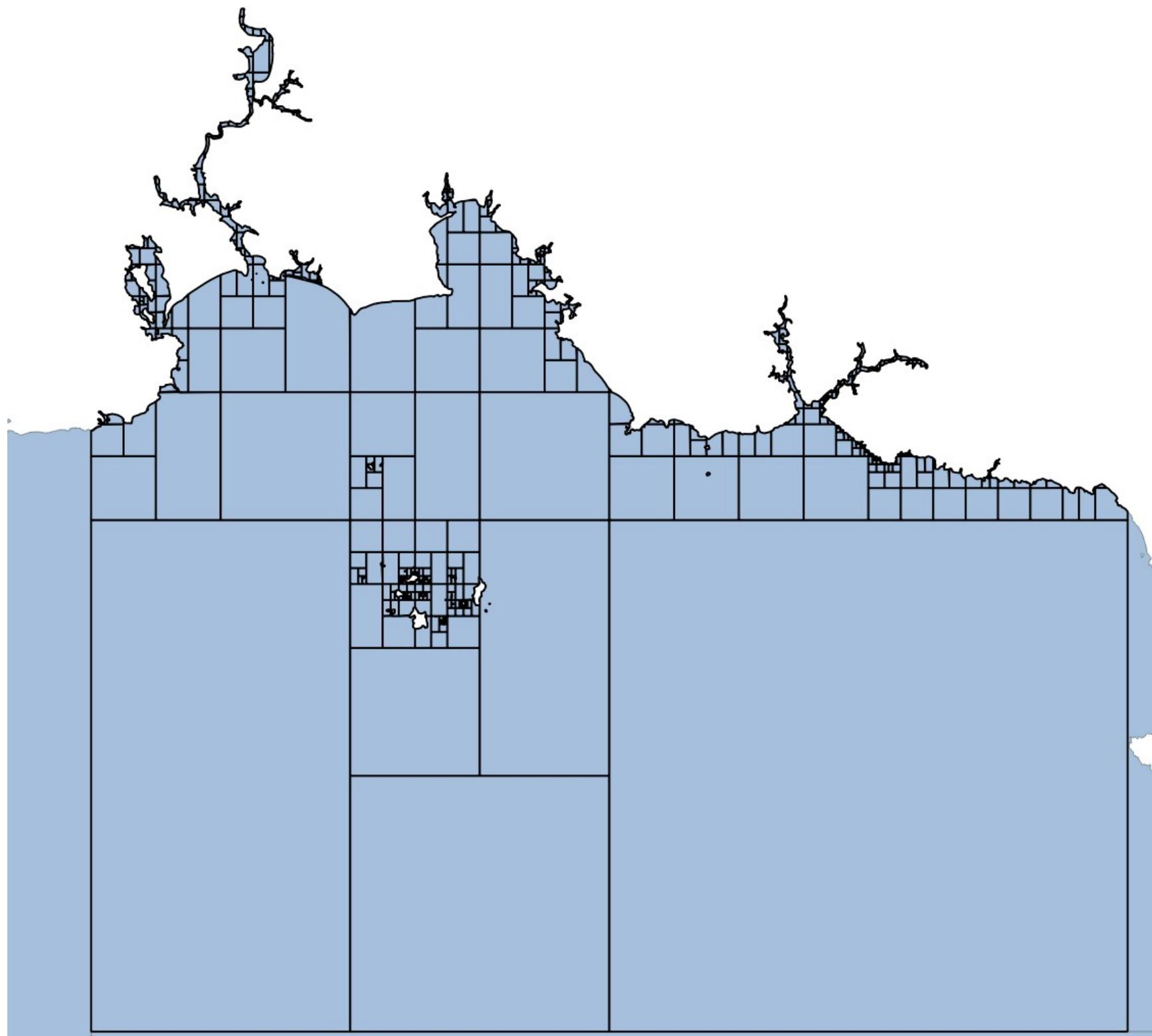
- › **Released earlier this month**
- › **Windows installer with :**
  - › **PostGIS 2.2, PgRouting 2.1.0, SFCGAL, OGR FDW, pgPointCloud...**
- › **New Features**
  - › **KNN Recheck, KNN Geography**
  - › **3D functions**
  - › **Temporal functions**
  - › **Raster improvement**
  - › **Other functions ( simplify, TWKB...)**





# Subdivide

```
SELECT
    ST_Subdivide(geom, 50) as geom
FROM
    water_polygon
WHERE
    gid = 16389;
```





# Simplify

```
SELECT
    gid
    , ST_SimplifyVW(geom, 300) as geom
FROM
    water_polygon
WHERE
    gid = 16389;
```



# TWKB

- > Binary format
- > Tiny Well Known Binary
- > With ids

**SELECT**

**ST\_AsTWKB(array\_agg(geom), array\_agg(gid))**

**FROM**

**mytable;**

**st\_astwkb**

---

-----  
  \x040402020400000202

# Raster



# Rasters

- > **Raster / vector analysis**
- > **New datatype**
  - > **Looks like geometry**
  - > **But for rasters**
- > **Multiresolution, multiband, tile coverage**
- > **Import/export (GDAL)**
- > **Functions**
  - > **Statistics, reprojection, edit, compute**
  - > **Vector/raster functions**
  - > **More & more functions & faster**

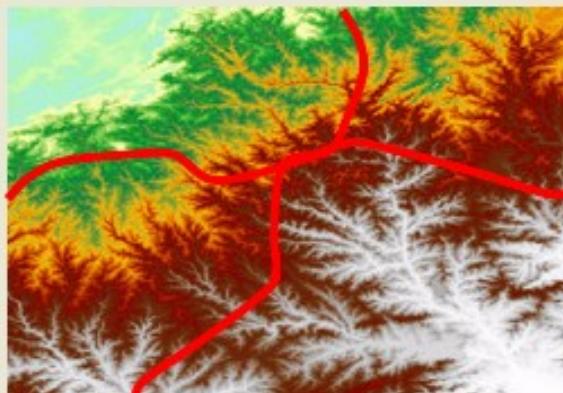
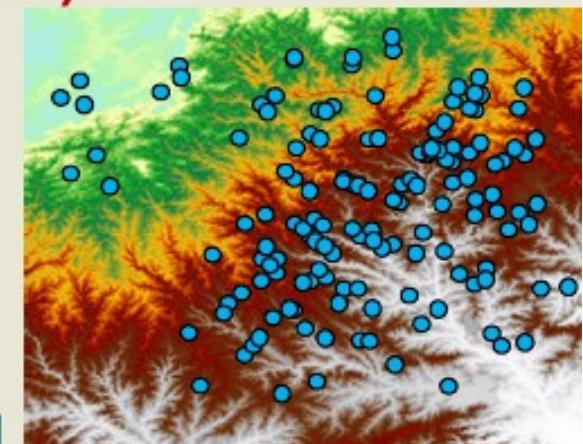
# Rasters

Extract ground elevation values for lidar points...

- `SELECT pointID, ST_Value(rast, geom) elevation  
FROM lidar, srtm WHERE ST_Intersects(geom, rast)`

Intersect a road network to extract  
elevation values for each road segment

- `SELECT roadID,  
(ST_Intersection(geom, rast)).geom road,  
(ST_Intersection(geom, rast)).val elevation  
FROM roadNetwork, srtm WHERE ST_Intersects(geom, rast)`



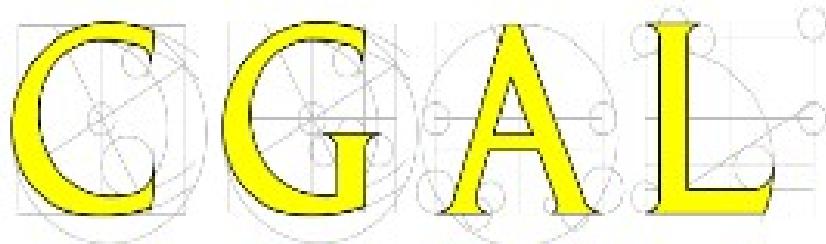
# 3D



# PostGIS 3D / SFCGAL

**SFCGAL**

=



+

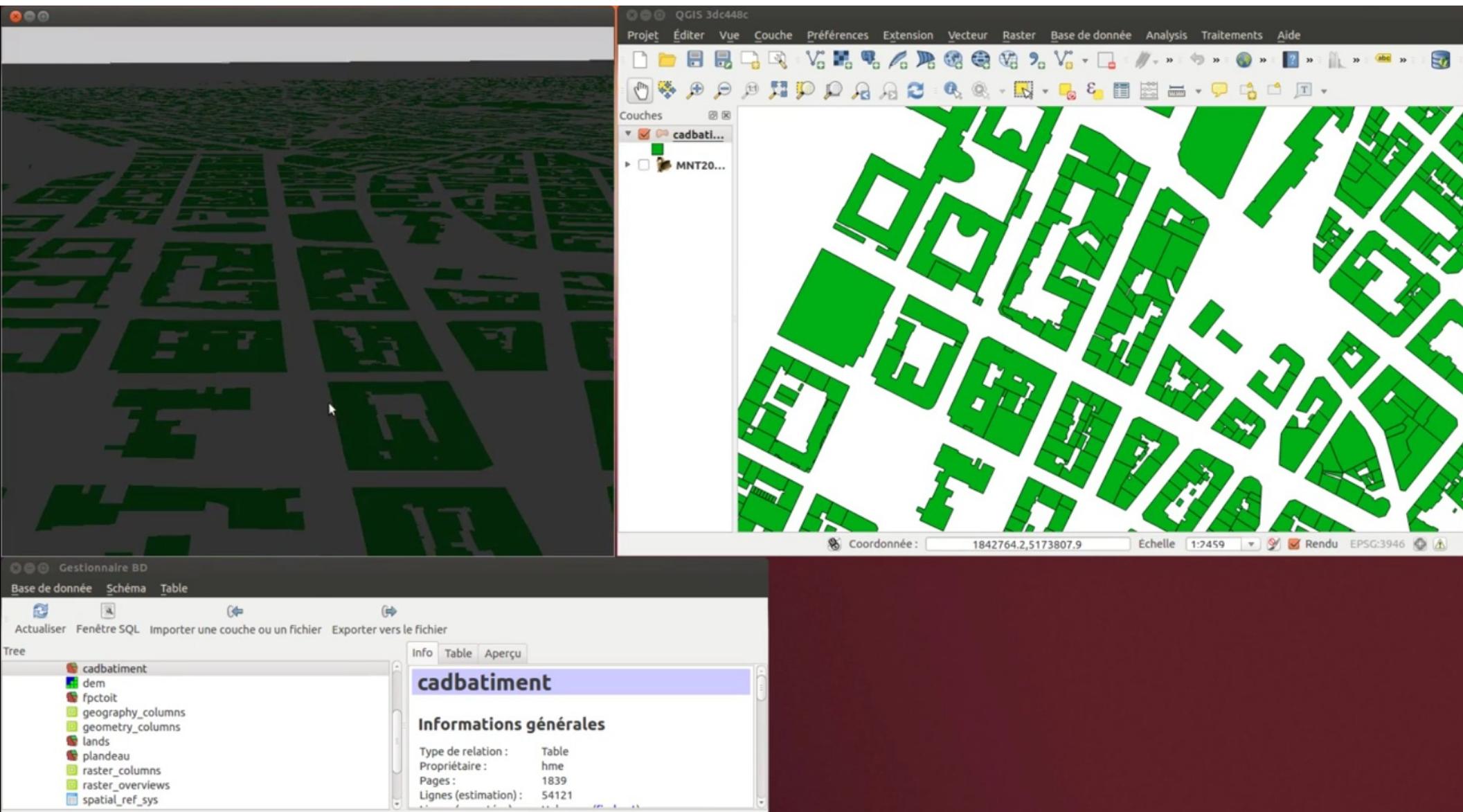


ISO 19107:2013

ISO 19125:2013

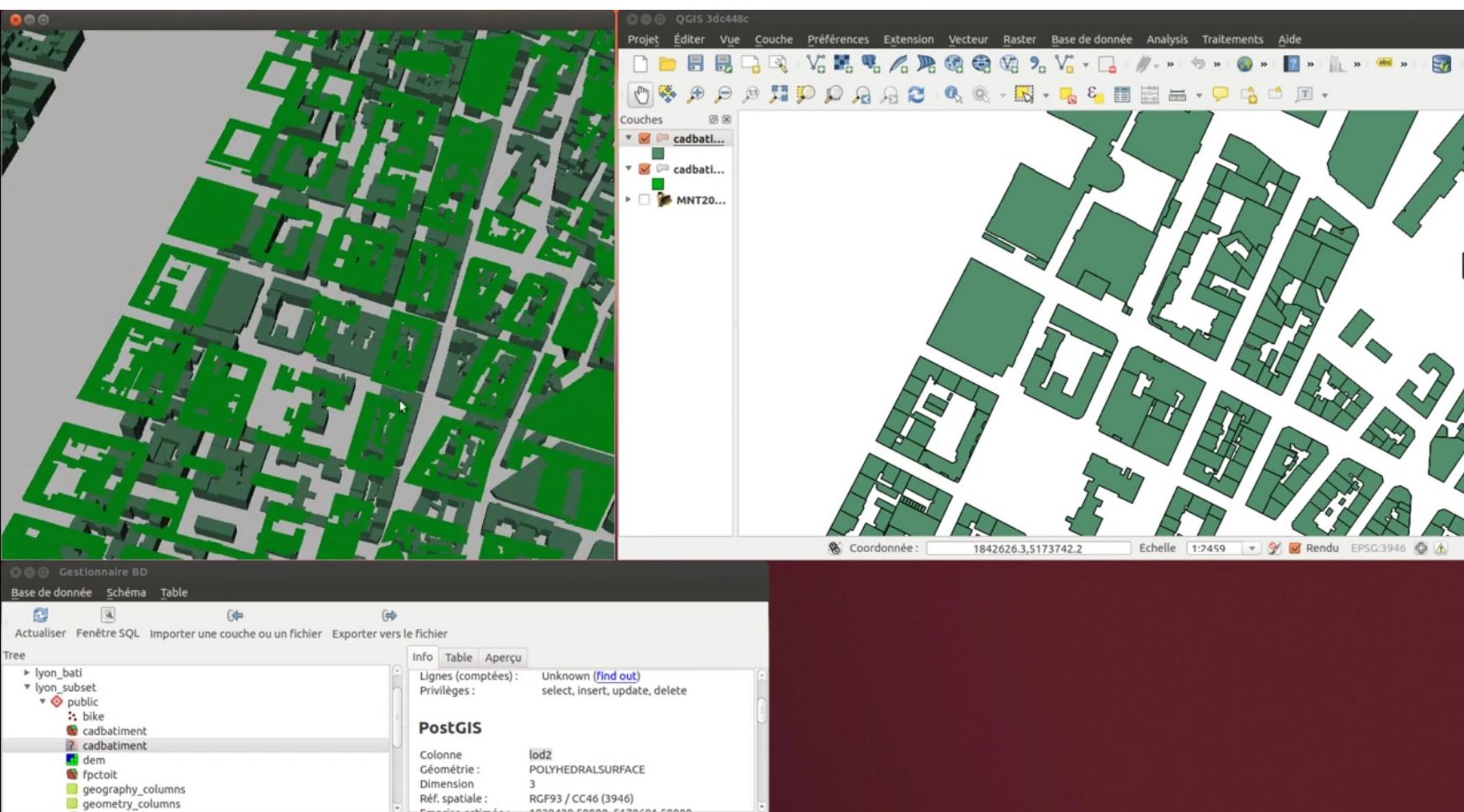


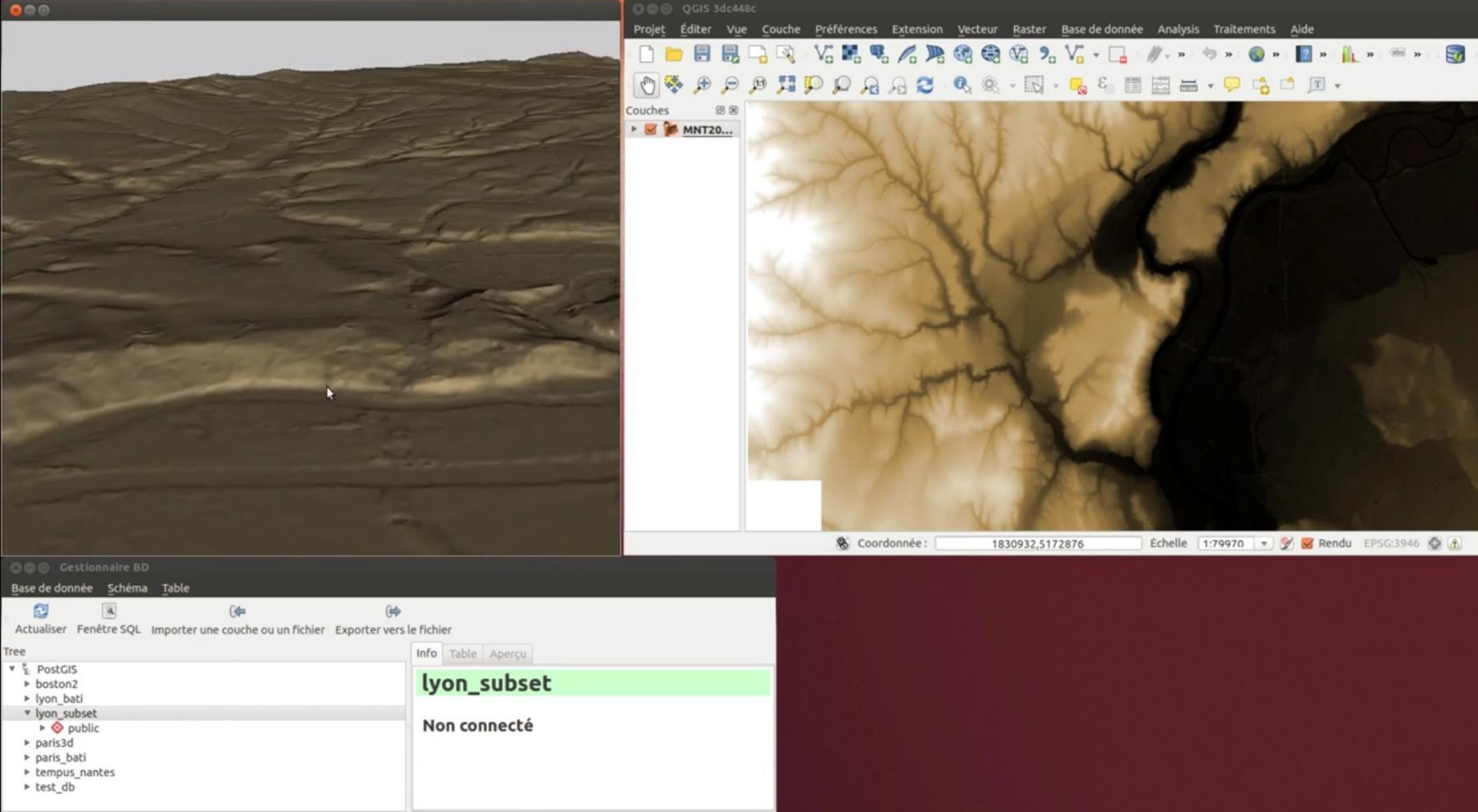
CGAL



```
ALTER TABLE  
  buildings  
ADD COLUMN  
  geom3d geometry('POLYHEDRALSURFACEZ',3946);
```

```
UPDATE  
  buildings  
SET  
  geom3d = ST_Extrude(ST_Force2D(geom), 0, 0, height);  
  
-- Create spatial indexes  
CREATE INDEX buildings_geom3d_idx  
  ON buildings USING GIST(lod2);
```

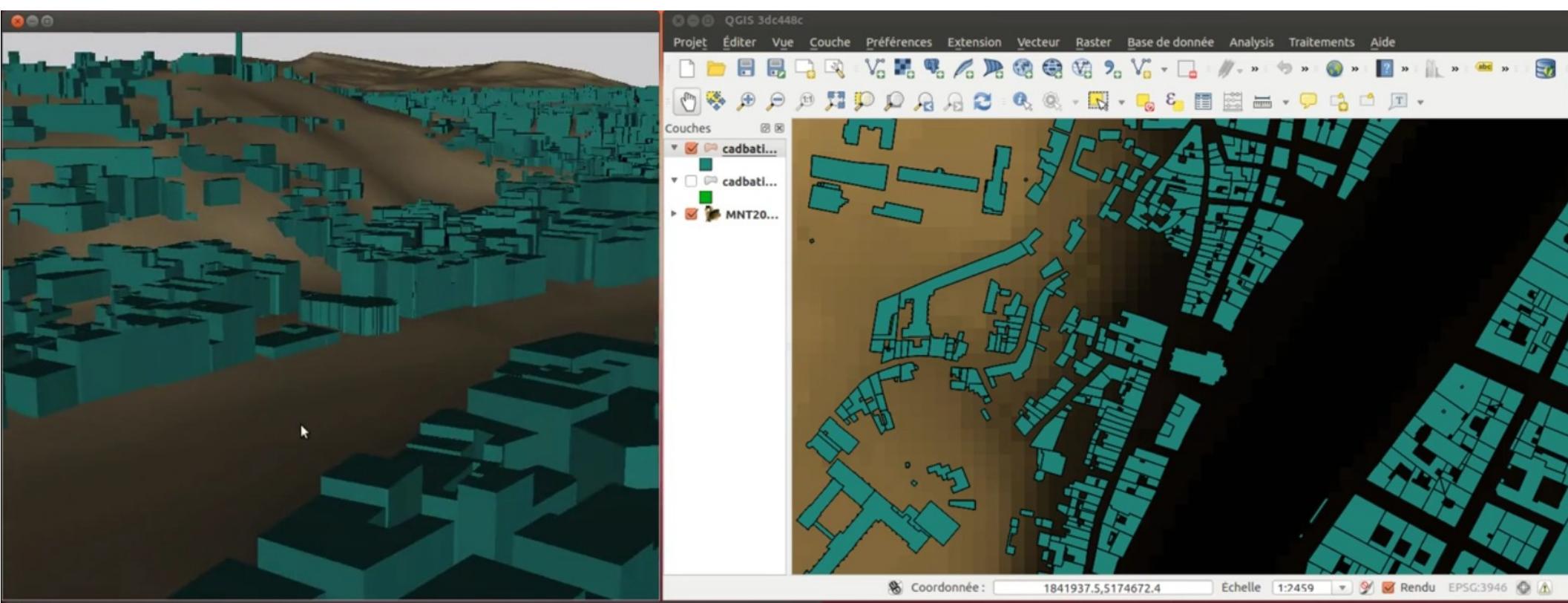




```
-- Compute altitude for each building
ALTER TABLE buildings ADD COLUMN altitude integer;

SELECT
    gid,
    -- Get the raster (elevation) value at the
    -- building's centroid point
    st_value(rast, st_centroid(geom)) as elevation
FROM
    buildings
ON
    -- Spatial join between raster and geometry
    ST_Intersects(rast, geom)
```

```
WITH hh AS
(
    ...
)
UPDATE
    buildings
SET
    -- Keep the altitude
    altitude = elevation
    -- Translate the buildings to their real heights
    , geom3d = ST_Translate(geom3d, 0, 0, elevation)
FROM
    hh
WHERE
    buildings.gid = hh.gid;
```



This screenshot shows a database management application window. At the top, there are tabs for 'Gestionnaire BD', 'Base de données', 'Schéma', and 'Table'. Below the tabs are buttons for 'Actualiser', 'Fenêtre SQL', 'Importer une couche ou un fichier', and 'Exporter vers le fichier'. The main area features a 'Tree' sidebar on the left listing database objects: 'lyon\_bati', 'lyon\_subset', 'public' (with 'bike', 'cadbatiment', 'cadbatiment'), 'dem', 'fpcotit', 'geography\_columns', and 'geometry\_columns'. The 'cadbatiment' object is selected and highlighted in purple. To the right, a detailed view of the 'cadbatiment' table is shown with tabs for 'Info', 'Table', and 'Aperçu'. The 'Info' tab displays the following information:

cadbatiment	
<b>Informations générales</b>	
Type de relation :	Table
Propriétaire :	hme
Pages :	17664
Lignes (estimation) :	54121

# Viewer : Horao

- › **OpenSceneGraph-based**
- › **Independant + QGIS plugin**
- › **Synchronization with QGIS**
- › **MNT, 3D, custom queries**
- › **[www.horao.net](http://www.horao.net)**



# PointCloud





**TOMTOM**

[www.tomtom.com](http://www.tomtom.com)

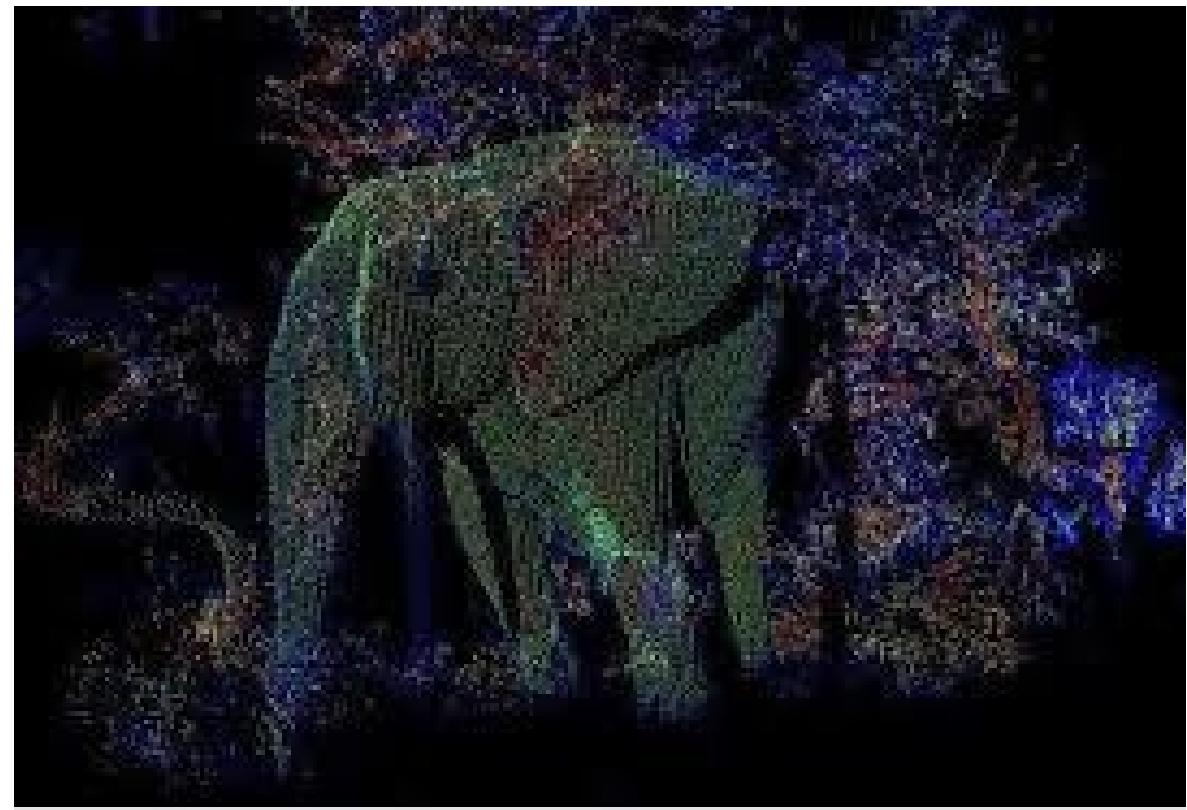
1-AJL-380

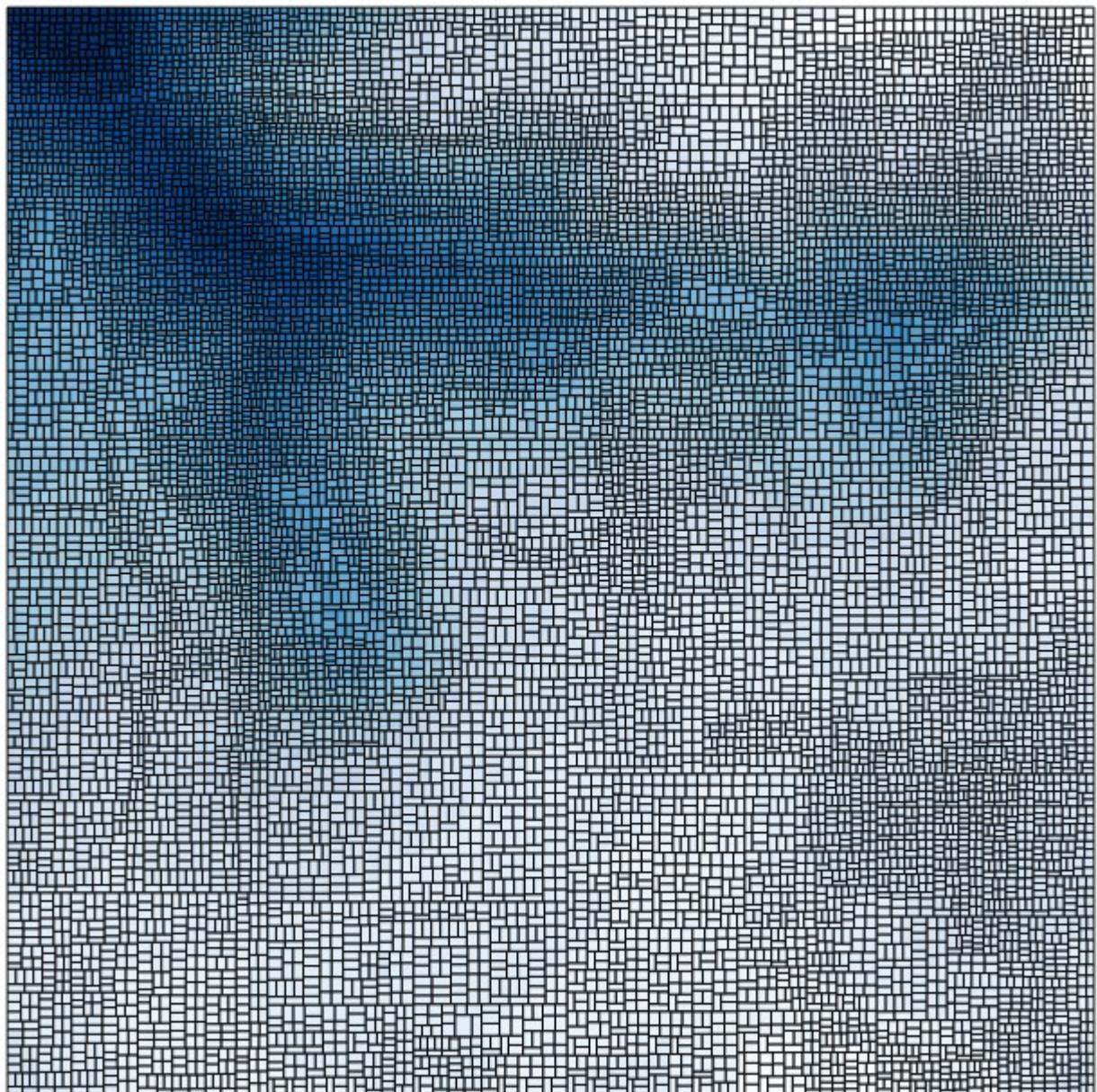
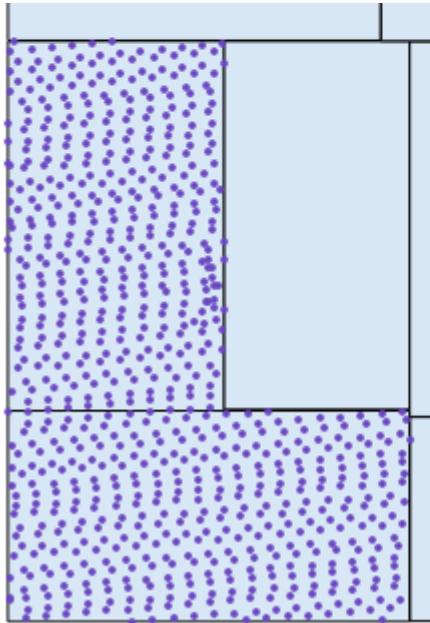
**TOMTOM**

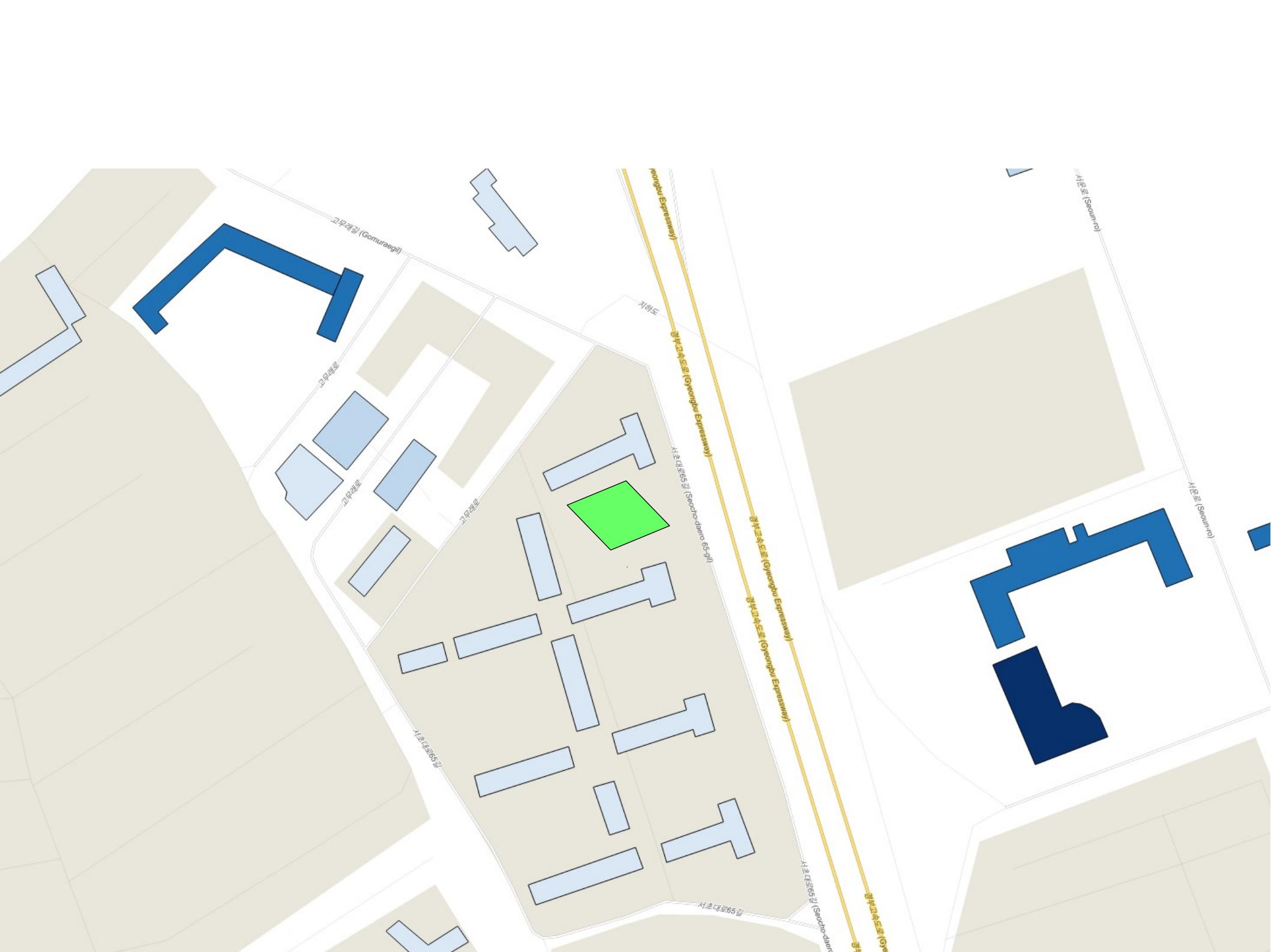
We're mapping your world.

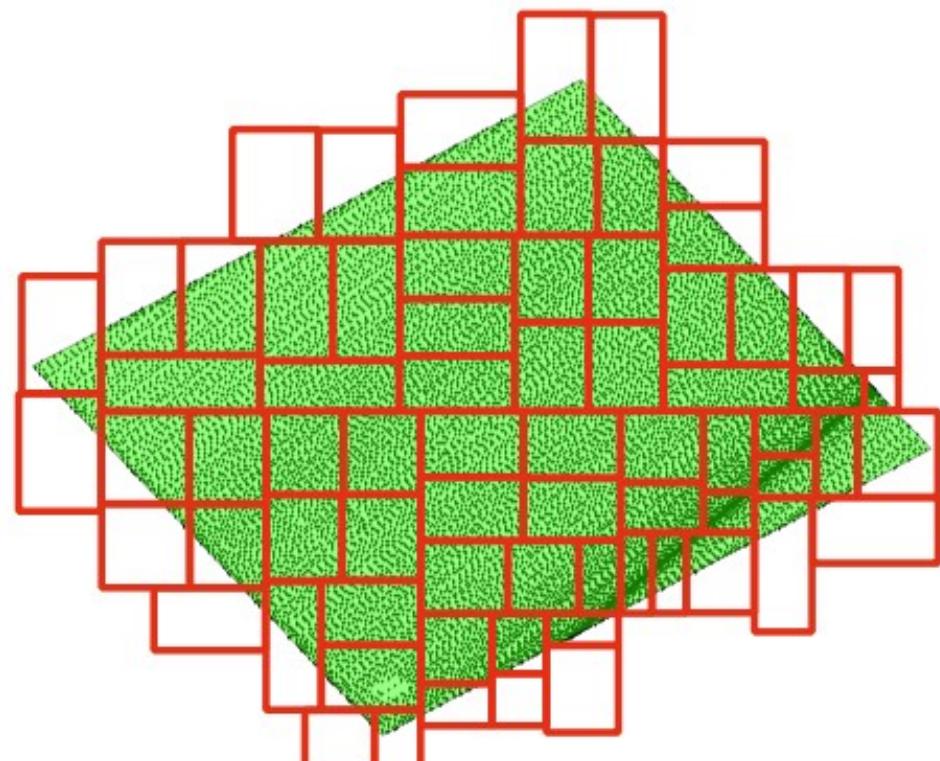
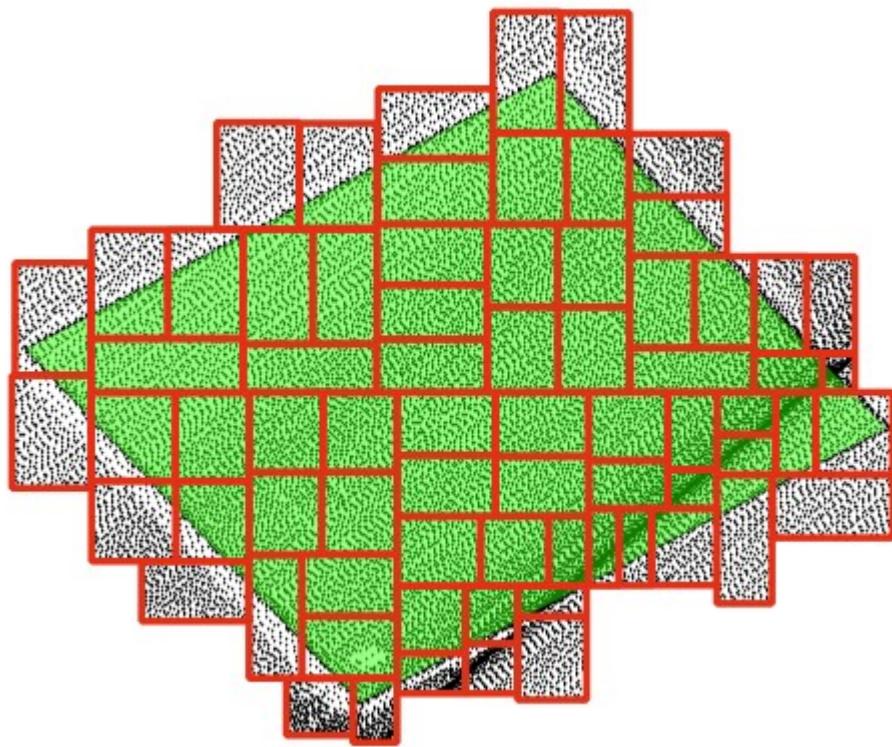
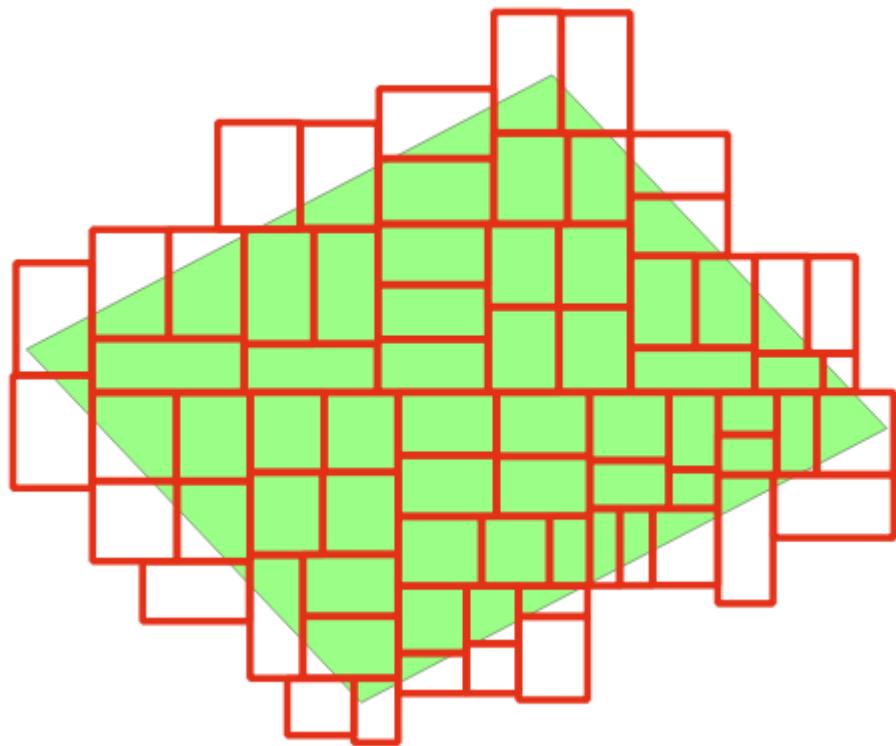
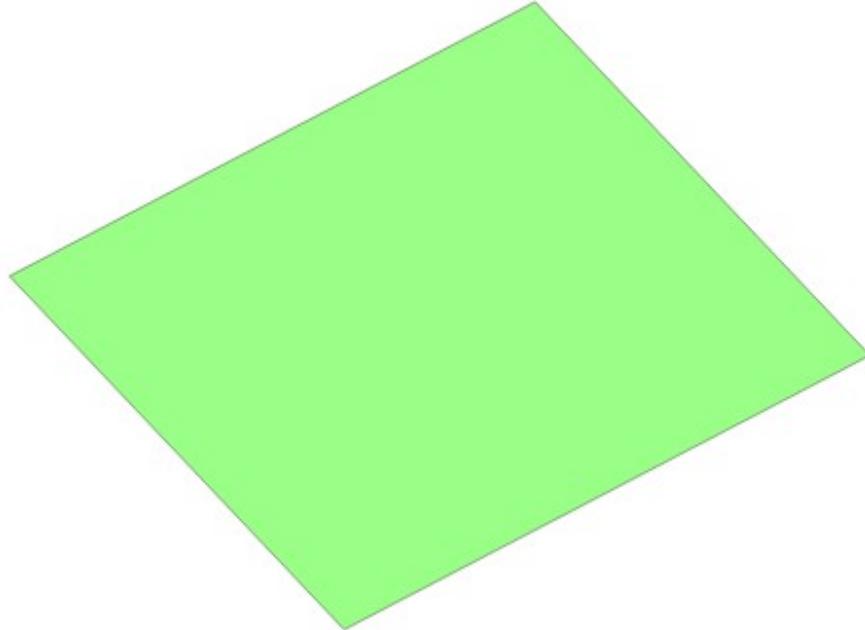
# PointCloud

- › **PostgreSQL extension + PostGIS**
- › **Paul Ramsey - Natural resources Canada**
- › **LIDAR - Huge point datasets**
- › **N-Dimensional**
- › **PDAL I/O**
  
- › **Point patches**
- › **Indexes**
- › **Functions**

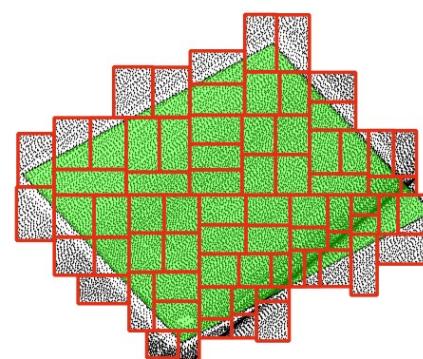
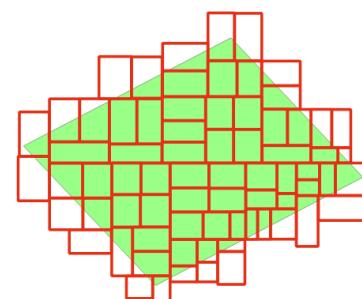
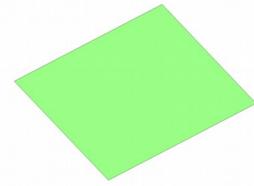








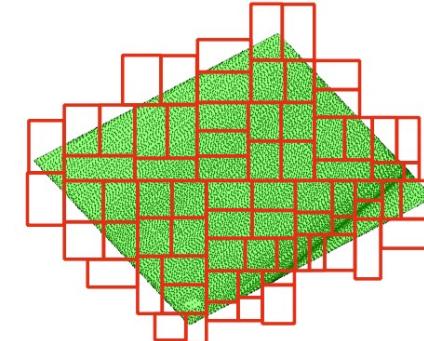
```
-- The building we are interested in
WITH building AS (
    SELECT geom FROM buildings WHERE buildings.gid = 81719
),
-- All the patches that intersect that building
patches AS (
    SELECT pa FROM pcpatches
        JOIN building ON PC_Intersects(pa, geom)
),
-- All the points in that patch
pa_pts AS (
    SELECT PC_Explode(pa) AS pts FROM patches
),
```



■ ■ ■

...

```
-- All the points in our one building
building_pts AS (
    SELECT pts FROM pa_pts
        JOIN building ON ST_Intersects(geom, pts::geometry)
)
-- Summarize those points by elevation
SELECT Avg(PC_Get(pts, 'z')) AS lidar_meters FROM building_pts;
```



# 441.075 meters !



# **Under fire !**





OTTAWA F.D.  
PHOTOGRAPHER

L13

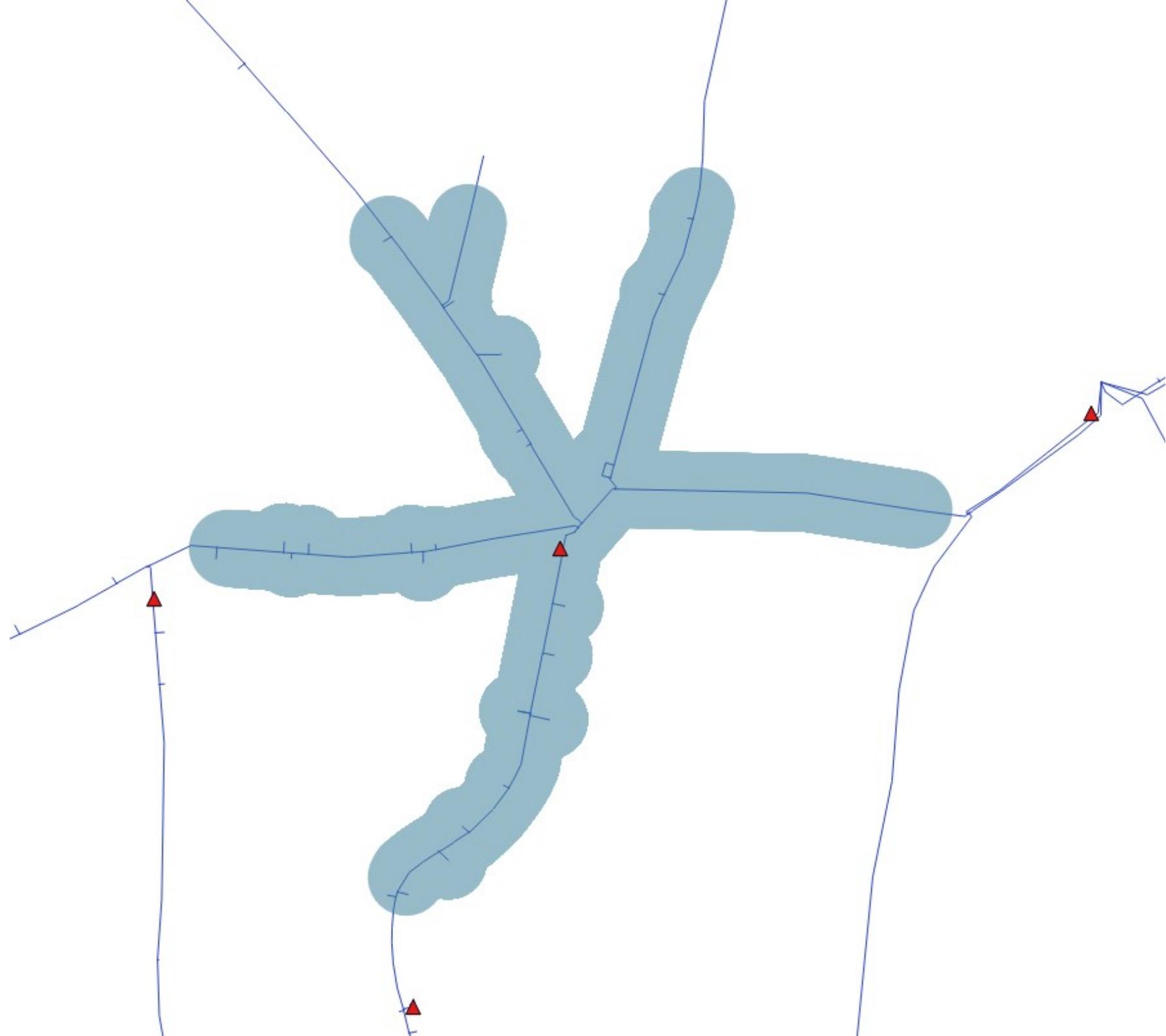
SUPPLY HOSE

# **Under fire !**

**Given a point on a map,  
find all zones where we can bring water**

- › **We have a topological pipe network**
- › **We must follow the streets ( = the pipes)**
- › **The fire hose is 220m**

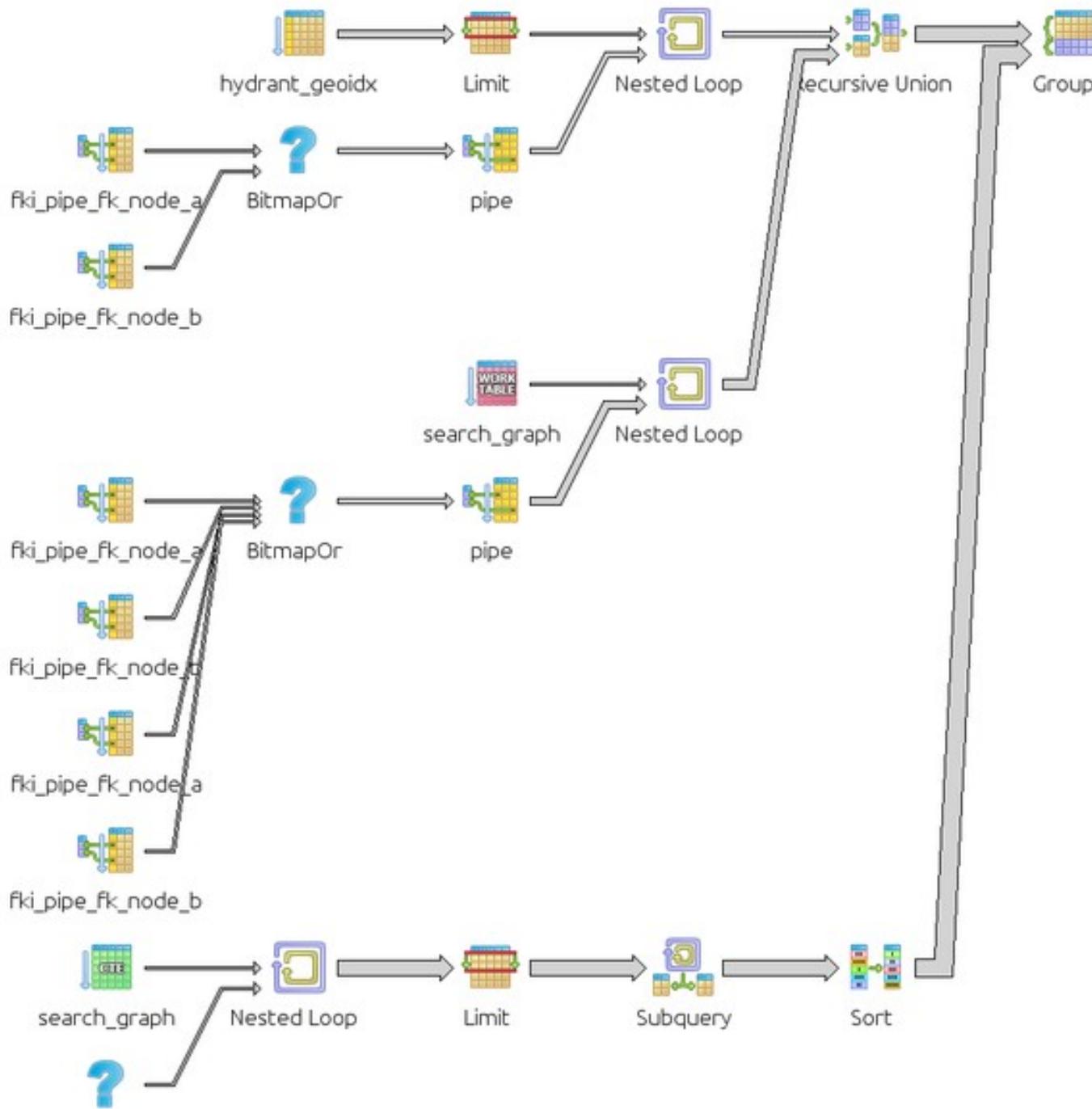




# **Under fire !**

- 1. Find nearest hydrant**
- 2. Get its connected pipes**
- 3. Follow the network for 200m**
- 4. At the end of reachable pipes,  
get only the right part < 200m**
- 5. Create a 20m zone around the paths**





Inly Scan using pipe\_pkey on pipe g

# 1. Nearest Hydrant

- › KNN-GIST search in PostgreSQL 9.1+
- › RECHECK in PG 9.5+
- › Use indexes !
- › Spatial nearest neighbors

```
select * from
  -- our hydrant
  qwat_od.hydrant as h
order by
  h.geometry <-> st_makepoint(845499.23,6473465.22, 2154)
limit 1;
```

- › NEW : NO need to refine for non-point geom

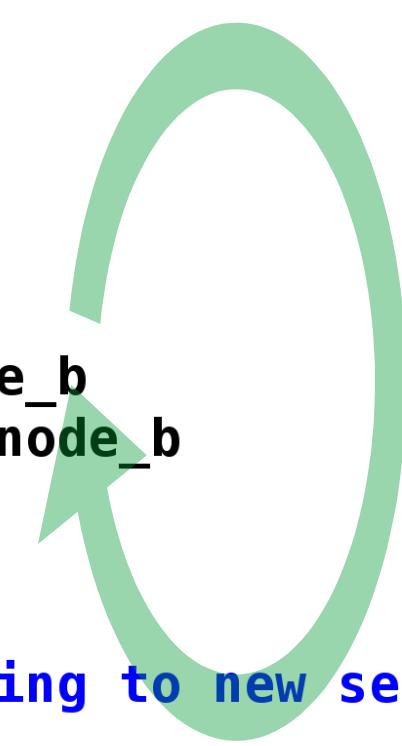
```
select
    g.id as gid
    , g.fk_node_a as nodea
    , g.fk_node_b as nodeb
    , g.geometry as geom
    , 1 as depth, ARRAY[g.id] as path
    , st_length(g.geometry) as length
    , false as cycle
    , st_length(g.geometry) > 200 as islast
    , true as forward_geom
from (
    -- our hydrant here
    ...
) as h
-- Get connected pipes
join
    qwat_od.pipe as g
    -- forward or backward
on h.fk_node = g.fk_node_a
or h.fk_node = g.fk_node_b
```

## 2. Get the pipes

# 3. Follow the network Recursive CTE !

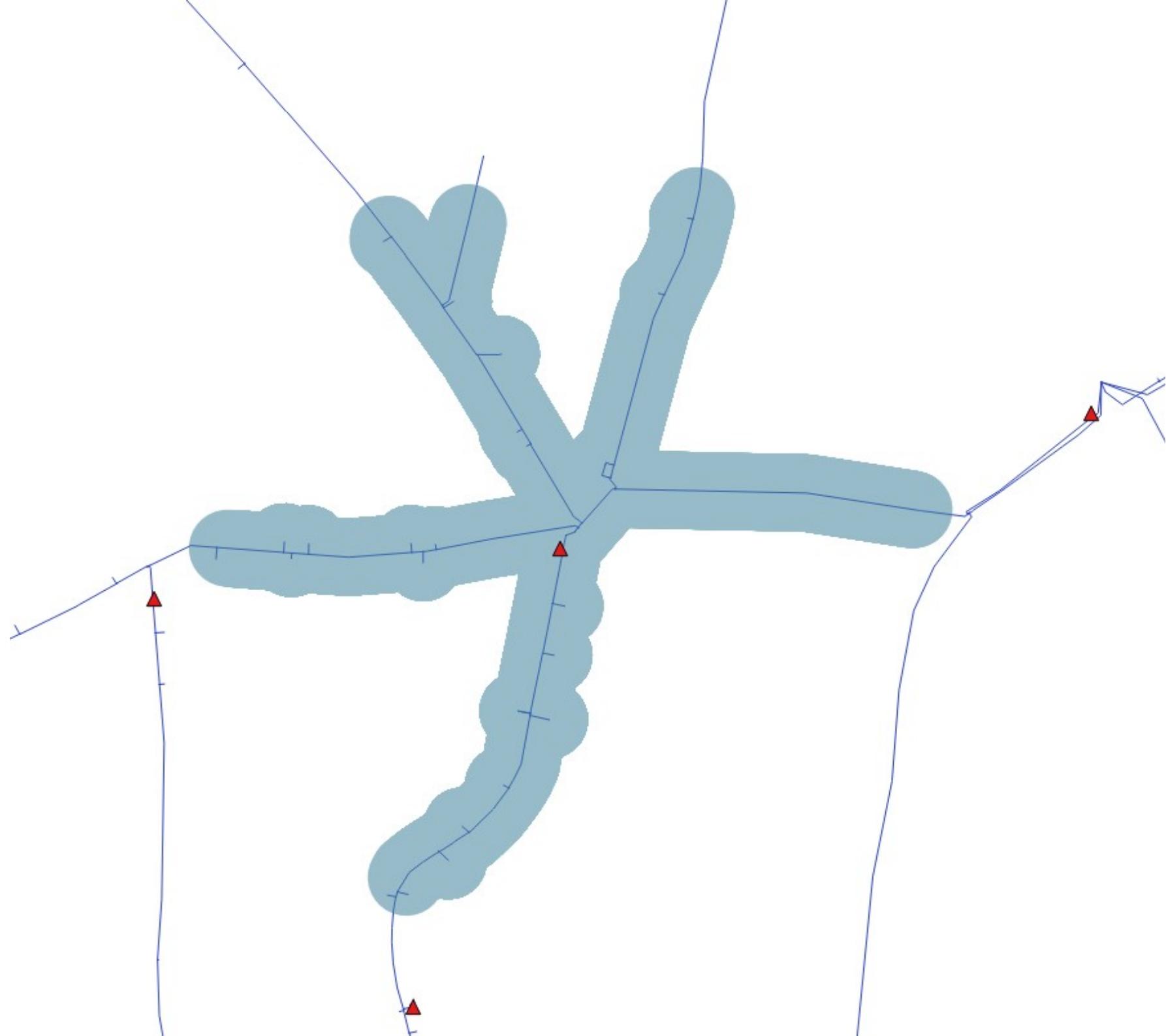
```
WITH RECURSIVE search_graph(...) AS
-- Initial dataset : pipes connected to our hydrant ( see #2 )
SELECT ... FROM qwat_od.hydrant JOIN qwat_od.pipe ...
-- Recursive dataset
UNION ALL
SELECT ...
FROM
    search_graph as sg
JOIN
    qwat_od.pipe as g
ON
    -- Previous dataset pipes are connected
    -- to other pipes one side or the other
    sg.nodeb = g.fk_node_a or sg.nodeb = g.fk_node_b
    or sg.nodea = g.fk_node_a or sg.nodea = g.fk_node_b
WHERE
    -- no loop
    and sg.gid <> g.id and not cycle
    -- when total computed length > 200, add nothing to new set
    and sg.length < 200
```

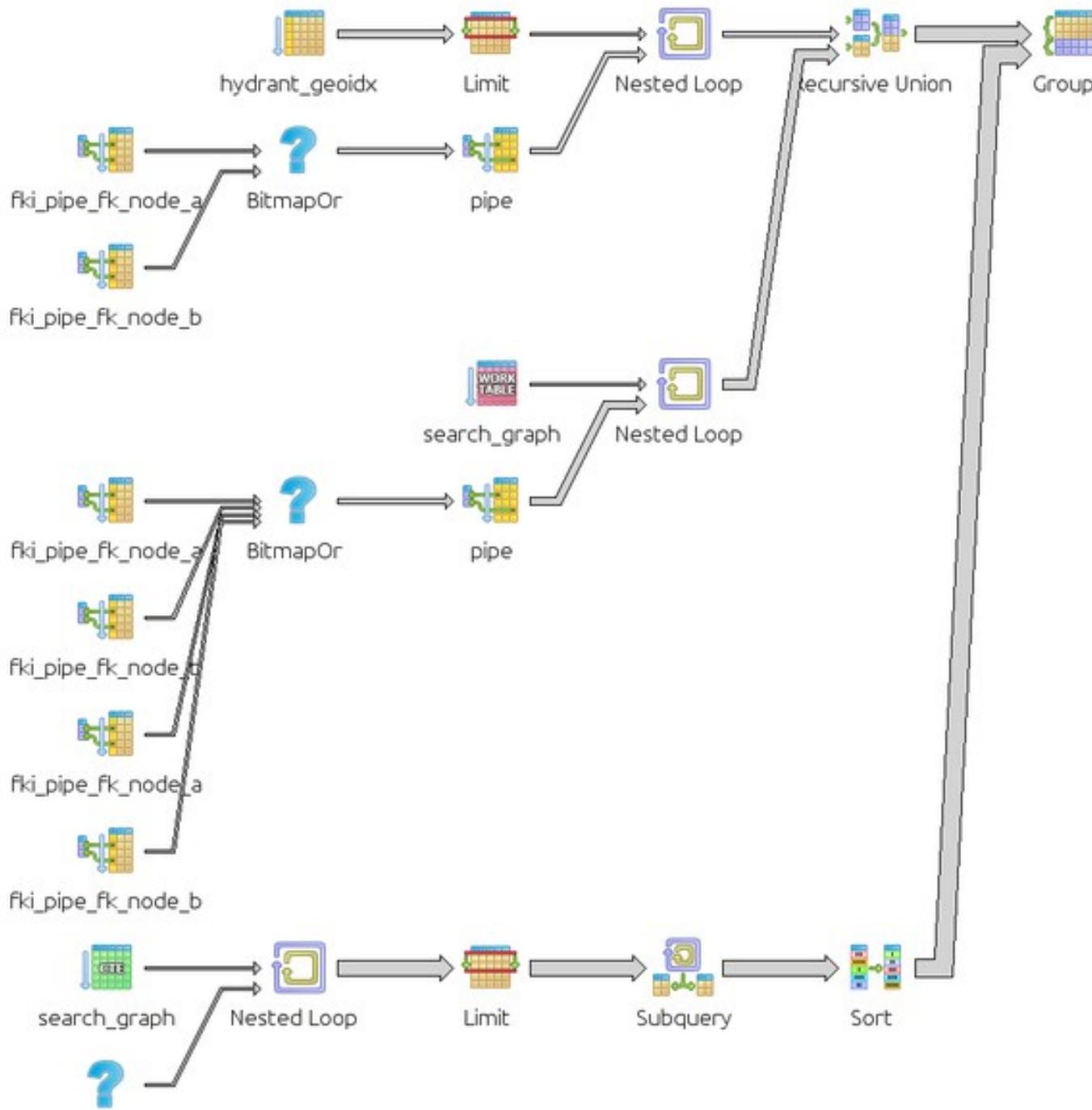
Computing cycle and length left as an exercice for the user



# 5. Buffer the paths

```
select sg.gid, sg.geom from (
select
    -- if pipe is part of path, buffer it
    , case
        when not islast then st_buffer(sg.geom, 20)
        -- if pipe is last of path, take only the portion to reach 200m
        -- and buffer it
        else st_buffer(
            st_linesubstring(sg.geom, 0,
                (200 + st_length(sg.geom) - sg.length) / st_length(sg.geom))
            , 20)
    end as geom
from
    search_graph as sg
-- limit the recursive search in case we
-- failed the stop criteria
limit 50000
```





Inly Scan using pipe\_pkey on pipe g

# Data from outer space



# **oracle\_fdw**

- › **FDW for Oracle database (Laurenz Albe)**
- › **NEW : Oracle Spatial support (Vincent Mora)**
- › **Native, R+W, Fast !**
- › **Points, lines & polygons**
- › **Combine with Materialized V.**
  
- › **Heterogeneous systems**
- › **Migration**
  
- › **ora2pg support**



# **ogr\_fdw**

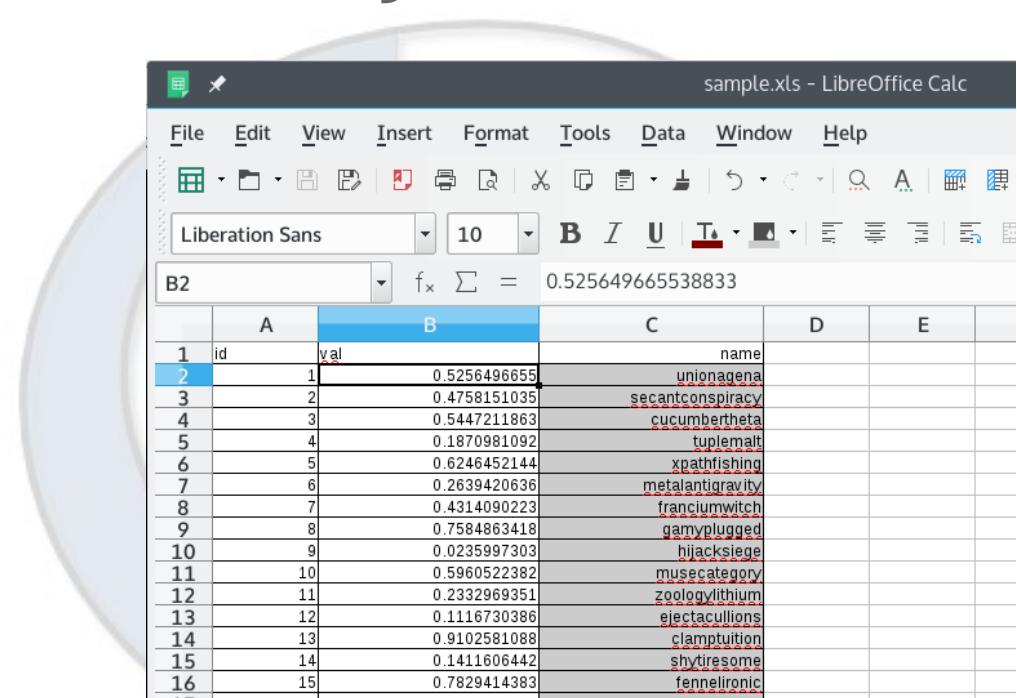
- › **OGR = vector data access**
- › **Foreign Data Wrapper**
- › **Geo / non-geo data**
- › **Lots of formats :**
  - › **Shapefile, SpatiaLite, WFS, GeoJSON...**



# ogr\_fdw

- > **OGR = vector data access**
- > **Foreign Data Wrapper**
- > **Geo / non-geo data**
- > **Lots of formats :**
  - > **Shapefile, SpatiaLite, WFS, GeoJSON...**

... Excel  
files !



The screenshot shows a LibreOffice Calc spreadsheet window titled "sample.xls - LibreOffice Calc". The menu bar includes File, Edit, View, Insert, Format, Tools, Data, Window, and Help. The toolbar has various icons for file operations, text styling, and data manipulation. The spreadsheet has a header row with columns A, B, C, D, and E. Column A is labeled "id", column B is "val", and column C is "name". The data consists of 20 rows of words and their corresponding values. Some values in column B are highlighted in red.

	A	B	C	D	E
1	id	val			name
2	1	0.525649665538833	unionagena		
3	2	0.4758151035	secantconspiracy		
4	3	0.5447211863	cucumbertheta		
5	4	0.1870981092	tuplemalt		
6	5	0.6246452144	xpathfishing		
7	6	0.2639420636	metalantigravity		
8	7	0.4314090223	franciumwitch		
9	8	0.7584863418	gamyplugged		
10	9	0.0235997303	hijacksiege		
11	10	0.5960522382	musecategory		
12	11	0.2332969351	zoologylithium		
13	12	0.1116730386	ejectacullions		
14	13	0.9102581088	clamptuition		
15	14	0.1411606442	shytiresome		
16	15	0.7829414383	fennelironic		
17	16	0.2554857663	epochcrook		
18	17	0.5174863243	bionicsexterior		
19	18	0.2345238051	triostrapping		
20	19	0.3626866782	crackedsounding		

```
./ogr_fdw_info -s /tmp/sample.xls -l Sheet1
```

```
CREATE EXTENSION ogr_fdw;
```

```
CREATE SERVER myserver
FOREIGN DATA WRAPPER ogr_fdw
OPTIONS (
    datasource '/tmp/sample.xls',
    format 'XLS');
```

```
CREATE FOREIGN TABLE sheet1 (
    fid integer,
    id integer,
    val real,
    name varchar )
SERVER myserver
OPTIONS ( layer 'Sheet1' );
```

```
pggis=# select * from sheet1 where val < 0.2;
```

fid	id	val	name
5	4	0.187098	tuplemalt
10	9	0.0235997	hijacksiege
13	12	0.111673	ejectacullions
15	14	0.141161	shytiresome
26	25	0.175975	broodbugs
28	27	0.0276925	lumberwhizzer
30	29	0.141428	sastor
36	35	0.0231349	pancakeplush
37	36	0.0174379	lithuaniancostly
48	47	0.131352	kaputalkane

(10 rows)

# Try it out



# docker-pggis

- › Docker container for latest database GIS stuff
- › Latest versions
  - › PG 9.5
  - › PostGIS 2.2 + SFCGAL + GEOS 3.5
  - › PointCloud master
  - › PDAL
  - › OGR\_FDW
- › Run everything in two lines
- › Based on phusion baseimage
- › Not for production !

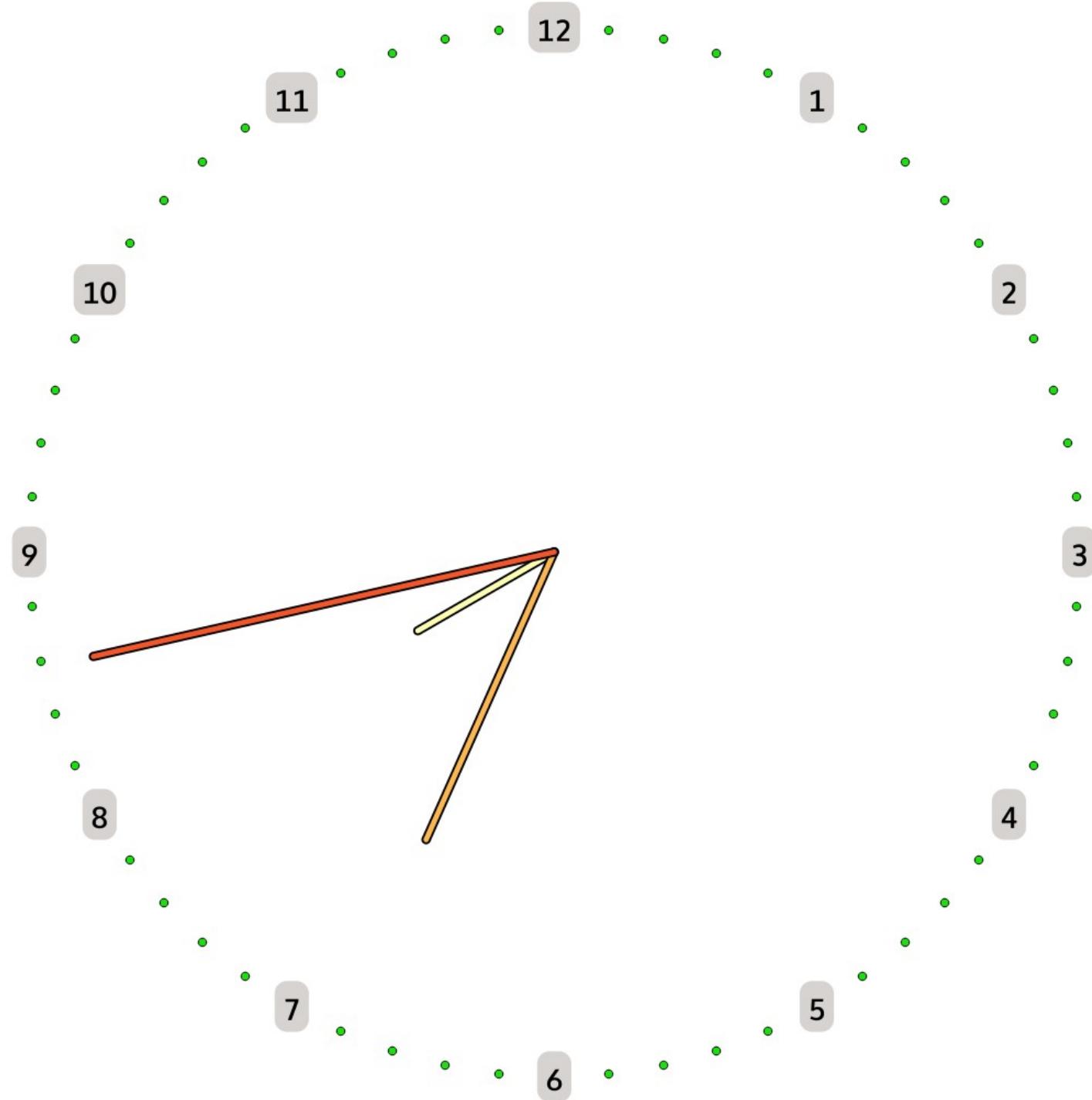


<https://github.com/vpicavet/docker-pggis>

docker build -t oslandia/pggis .  
docker run --rm -P --name pggis\_test oslandia/pggis

# PostGIS Clock





```
create or replace view ticks as
select
    m as id
    , case when m % 5 = 0 then 'h' else 'm' end as typ
    , m / 5 as h
    , m as m
    , st_makeline(
        cos(m::double precision * -pi() / 30.0 + pi() / 2)
        , sin(m::double precision * -pi() / 30.0 + pi() / 2)
    )
from
    generate_series(1, 60) as m;
```



```
create or replace view clock as
with hms (id, unit, length, nb, val) as (
    values (1, 'h', 0.3, 12, extract('hour' from current_time))
    , (2, 'm', 0.6, 60, extract('minute' from current_time))
    , (3, 's', 0.9, 60, extract('second' from current_time))
)
select
    hms.id
    , st_makeline(
        st_makepoint(0, 0)
        , st_makepoint(
            hms.length * cos(val::double precision *-2*pi() / nb + pi() / 2)
            , hms.length * sin(val::double precision *-2*pi() / nb + pi() / 2)
        )
    ) as geom
from
    hms;
```

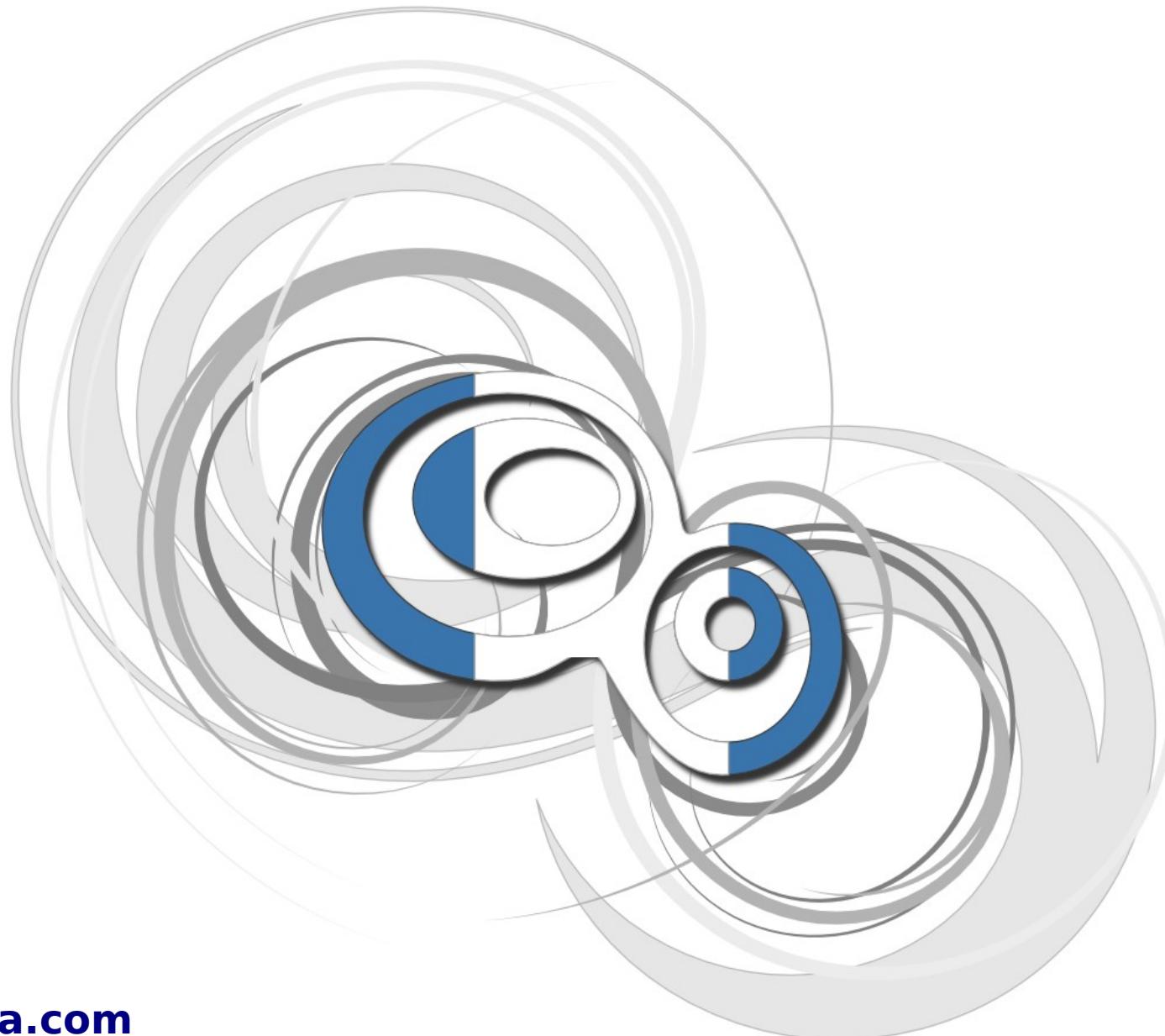


# Thank you



**vincent.picavet@oslandia.com**  
@vpicavet

OSLANDIA



**www.oslandia.com**

@Oslandia\_en / @Oslandia\_Team