PostGIS

Vincent Picavet - Oslandia -
www.oslandia.fr

SIG, principes

SIG, principes

Capturer

Créer

Stocker

Analyser

Partager

Visualiser

des données
géolocalisées

Base de données spatiale

Base de données spatiale

Stocker – géométrie + attributs

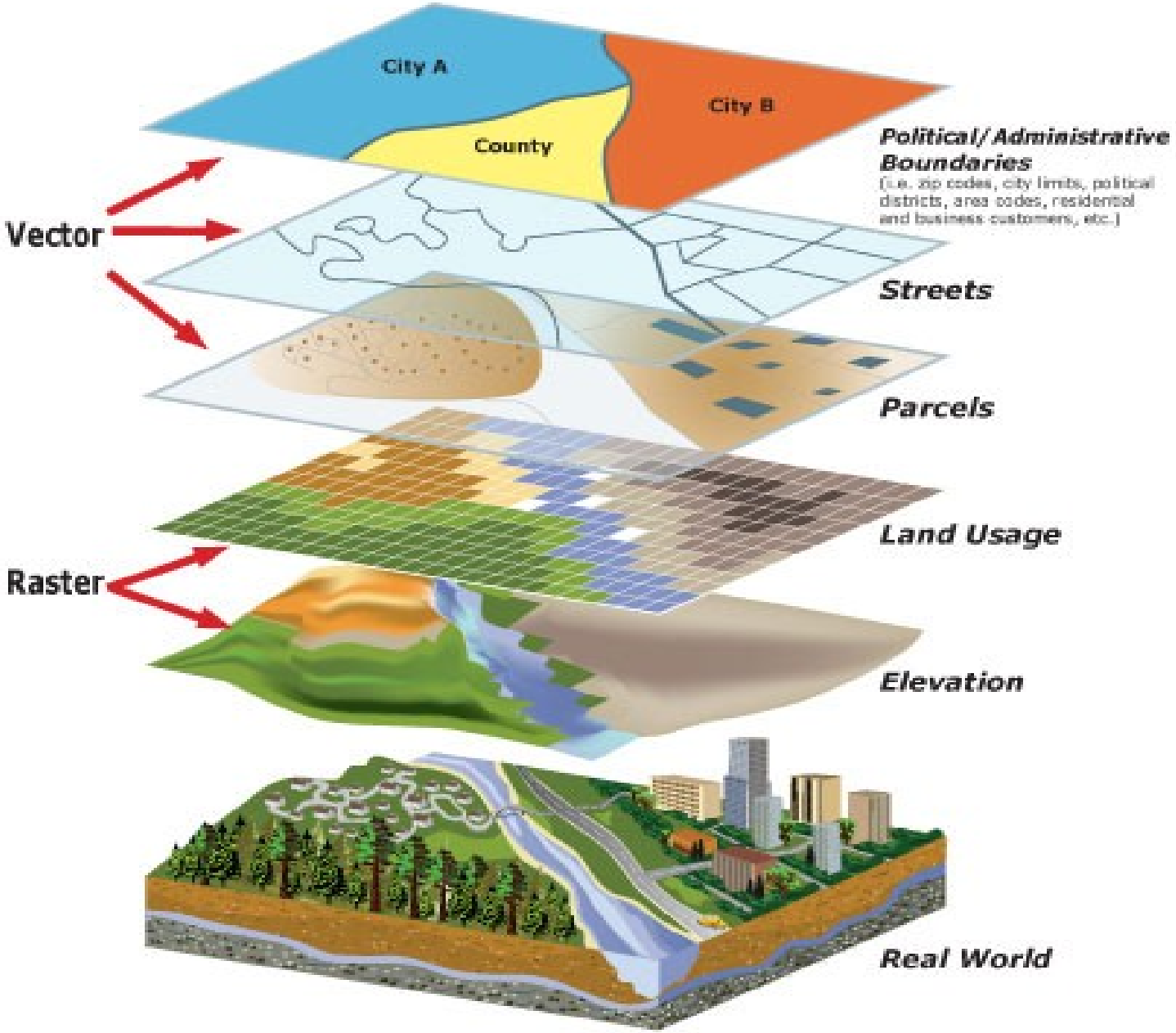
Requêter – spatial + attribut

Aller vite ! => indexation

Standards (OGC SFS ou ISO SQL/MM)

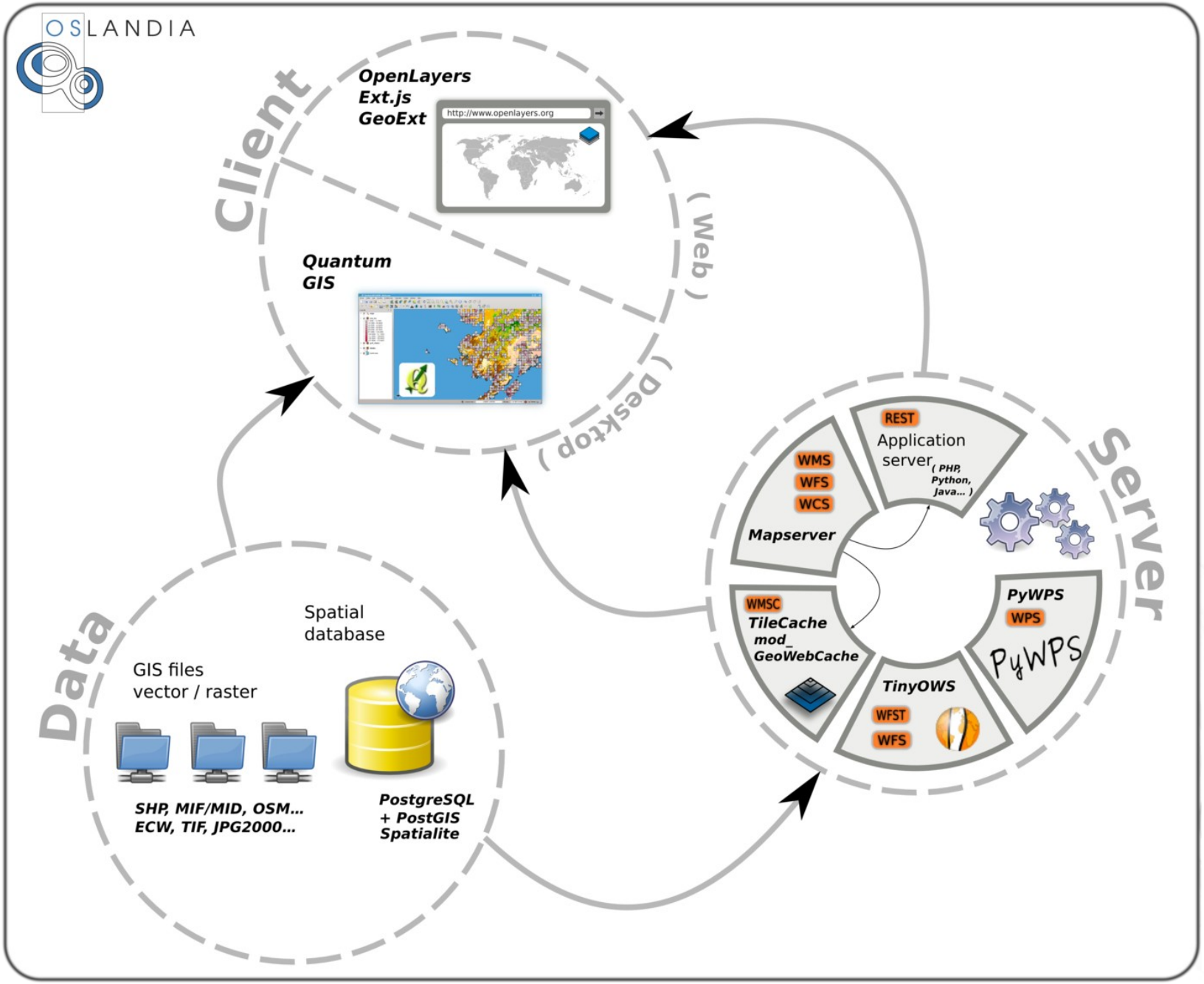
Gros volumes de données (plusieurs To)

Base de données spatiale – type de données



Intégration dans les architectures SIG

Intégration Archi SIG



Le projet PostGIS

PostGIS - Présentation

http://www.postgis.org

Version actuelle 2.1.4

Plugin PostgreSQL en C

Utilisé dans de très nombreux projets

- **Références prestigieuses**
- **Communauté large et technique**

OGC SFS (Simple Feature for SQL)
SQL/MM

Projet OSGeo

PostGIS - Historique

2001	Première version alpha
2003	Version 0.8 - Utilisation en production
2005	Version 1.0 - Réécriture du coeur et LWGEOM (OGC SFS 1.1)
2006	Version 1.2 - Cap sur ISO SQL/MM (Curves, préfixes ST_...)
2009	Version 1.4 - Création d'un PSC et entrée OSGeo
2010	Version 1.5
2012	Version 2.0 – Support Tin et Polyhedral (3D), Raster
2013	Version 2.1

PostGIS - Architecture et librairies

PostGIS : plugin PostgreSQL

Proj4

pour projection

GEOS

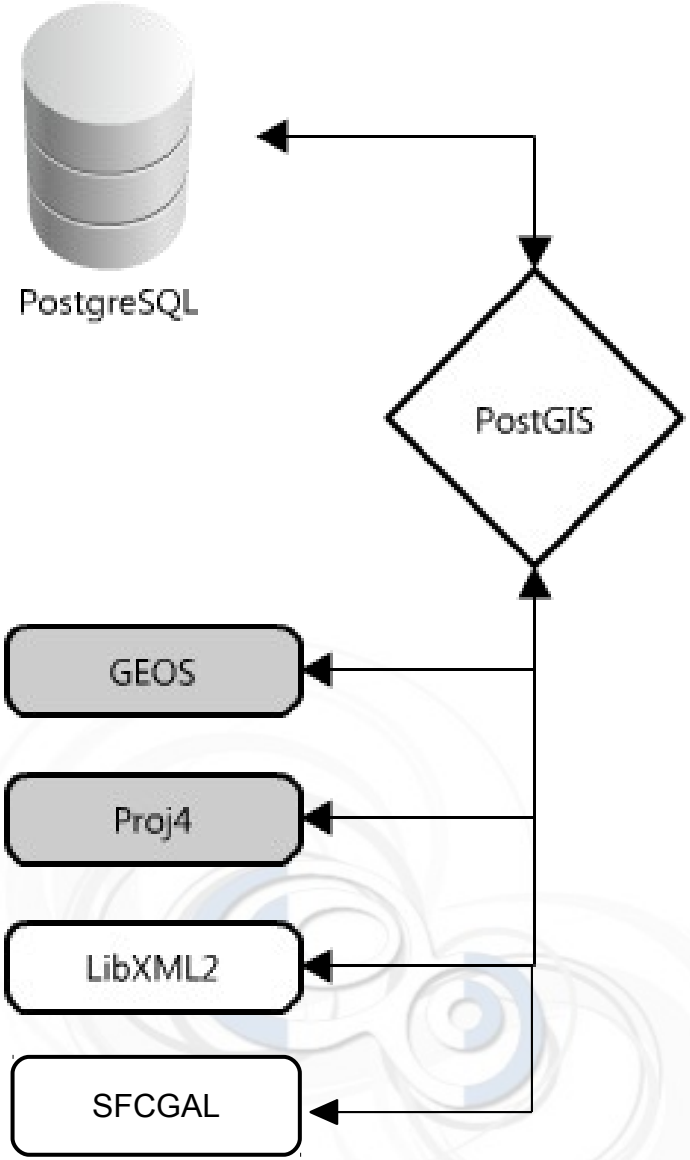
opérateurs spatiaux

LibXML2

pour le XML (GML/KML)

SFCGAL

analyse 3D



Représentation et stockage de géométrie

Géométries : représentation, stockage

- **Geometry (ou HEWKB)**
  - Stockage natif en base
  - Format binaire encodage hexadécimal
- **WKT (Well Known Text)**
  - Représentation textuelle
- **Dimensions**
  - 2D, 3D, ou 4D
- **Identifiant de projection (SRID)**

Lat/Lon : à quoi ça sert ?

Recherche des 10 plus proches restaurants en GeoJSON

```
with index_query as (
  select
    st_distance(way, 'SRID=900913;POINT(537000 5742000)') as distance,
    name,
    way
  from planet_osm_point
     where amenity = 'restaurant' and name is not null
  order by way <#> 'SRID=900913;POINT(537000 5742000)' limit 100
)
select
   distance,
   name,
   st_asgeojson(way)
from index_query
order by distance limit 10;
```

Recherche des 10 plus proches restaurants en GeoJSON

	distance double precision	name text	st_asgeojson text
1	168.462107905611	La petite table des Nuits	{"type":"Point","coordinates":[536831.63,5741994.43]}
2	168.671700353034	Daniel et Denise	{"type":"Point","coordinates":[537167.24,5742021.93]}
3	194.197821048509	Al Dente	{"type":"Point","coordinates":[537175.11,5741916.04]}
4	217.798582640138	La Morille	{"type":"Point","coordinates":[537207.51,5741933.85]}
5	260.85059037672	La Crise	{"type":"Point","coordinates":[537217.13,5741855.44]}
6	312.777100983944	New Delhi	{"type":"Point","coordinates":[537292.35,5741888.82]}
7	314.244424612237	Le Coquemar	{"type":"Point","coordinates":[536773.4,5742217.72]}
8	323.875102161495	Peshawar	{"type":"Point","coordinates":[537206.47,5741750.47]}
9	341.372962608132	Pierre & Martine	{"type":"Point","coordinates":[537285.86,5741813.4]}
10	344.388016196797	Bombay Palace	{"type":"Point","coordinates":[537301.21,5741833.04]}

Charger de la donnée

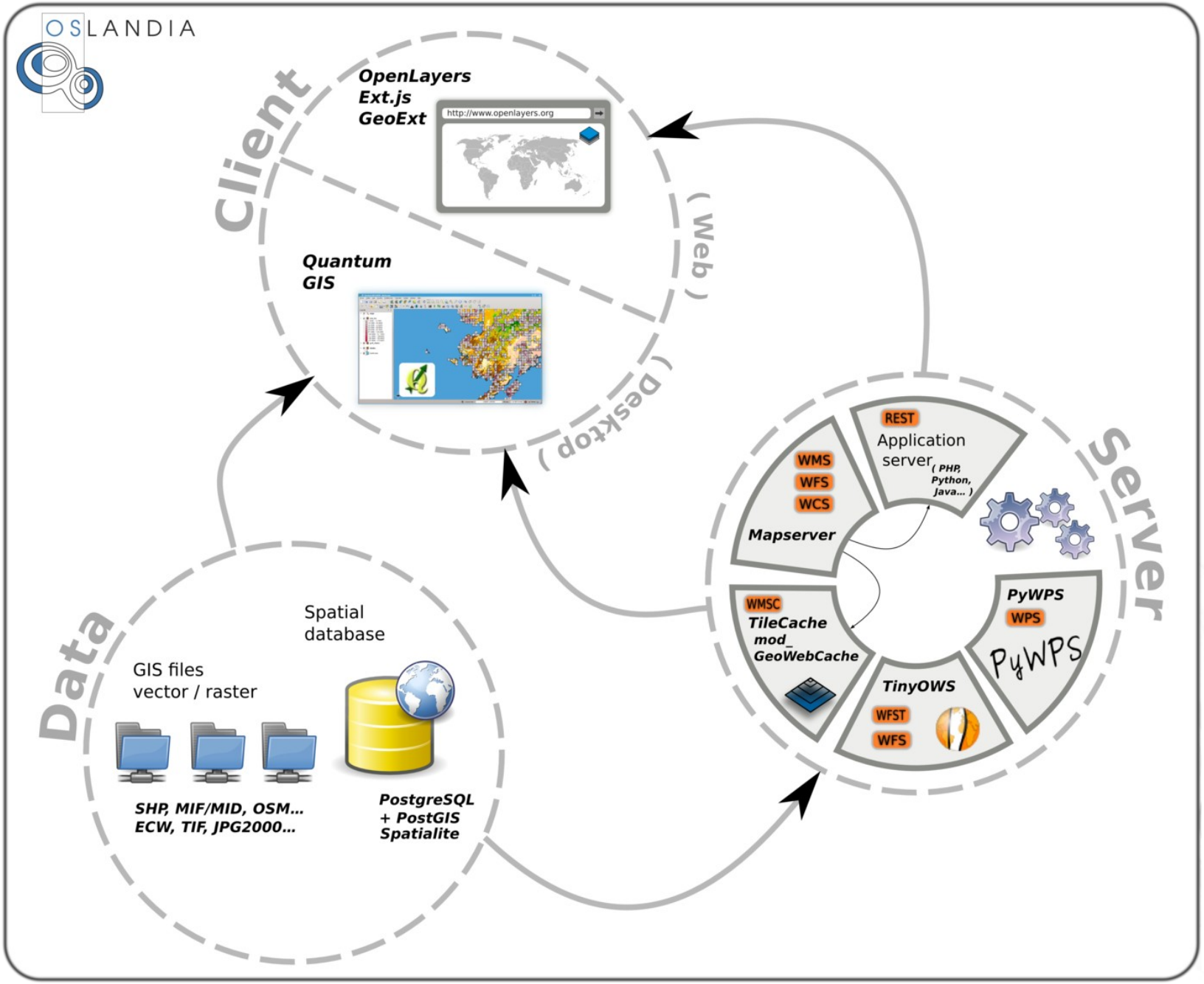
Charger de la donnée

- Shapefile GUI (shp2pgsql)
- GDAL/OGR
- OSM (osm2pgsql, osmosis...)

```
osm2pgsql -d osm -U user rhone-alpes-latest.osm.pbf
```

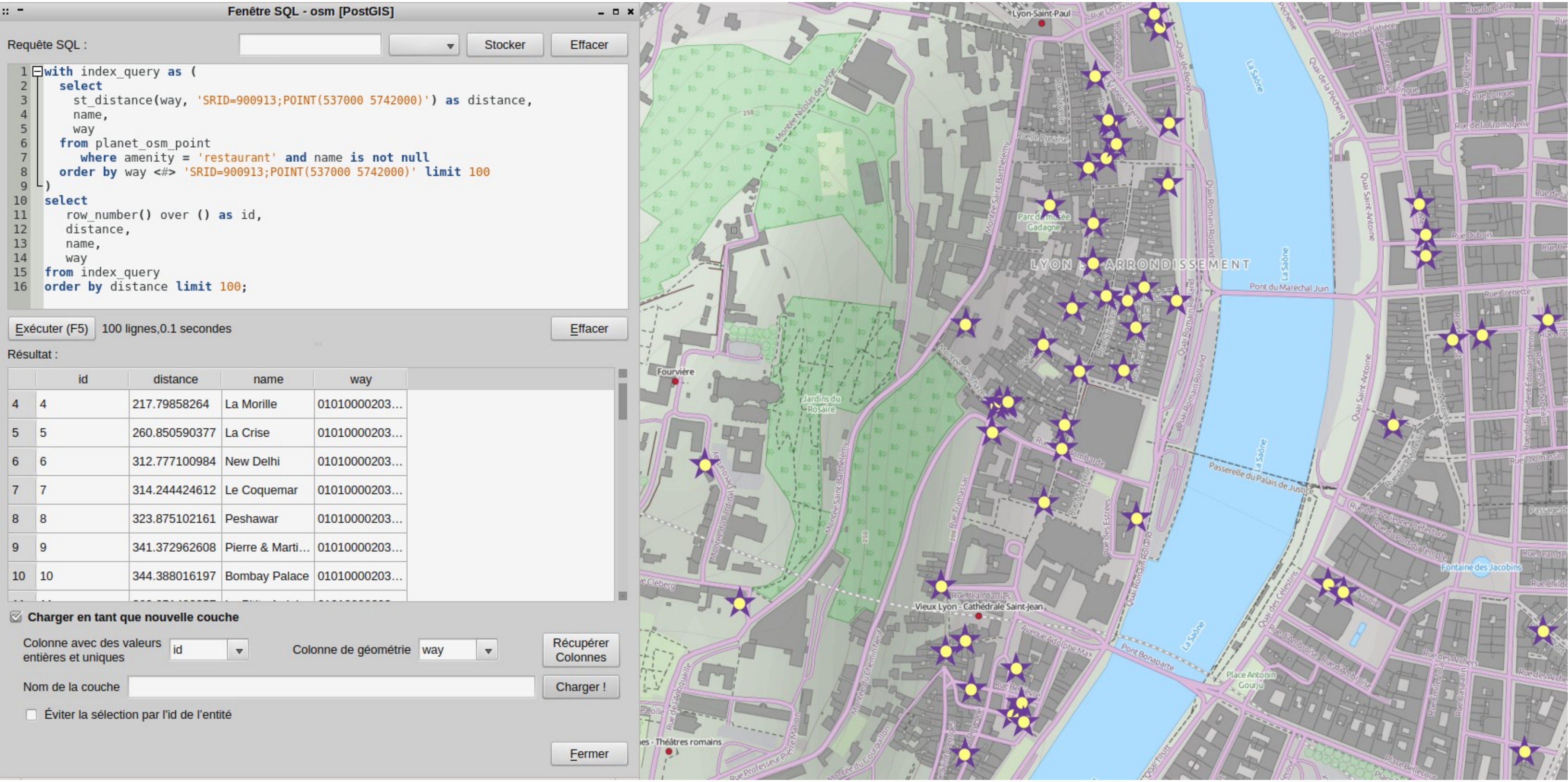
Utiliser PostGIS

Utiliser PostGIS - rappel



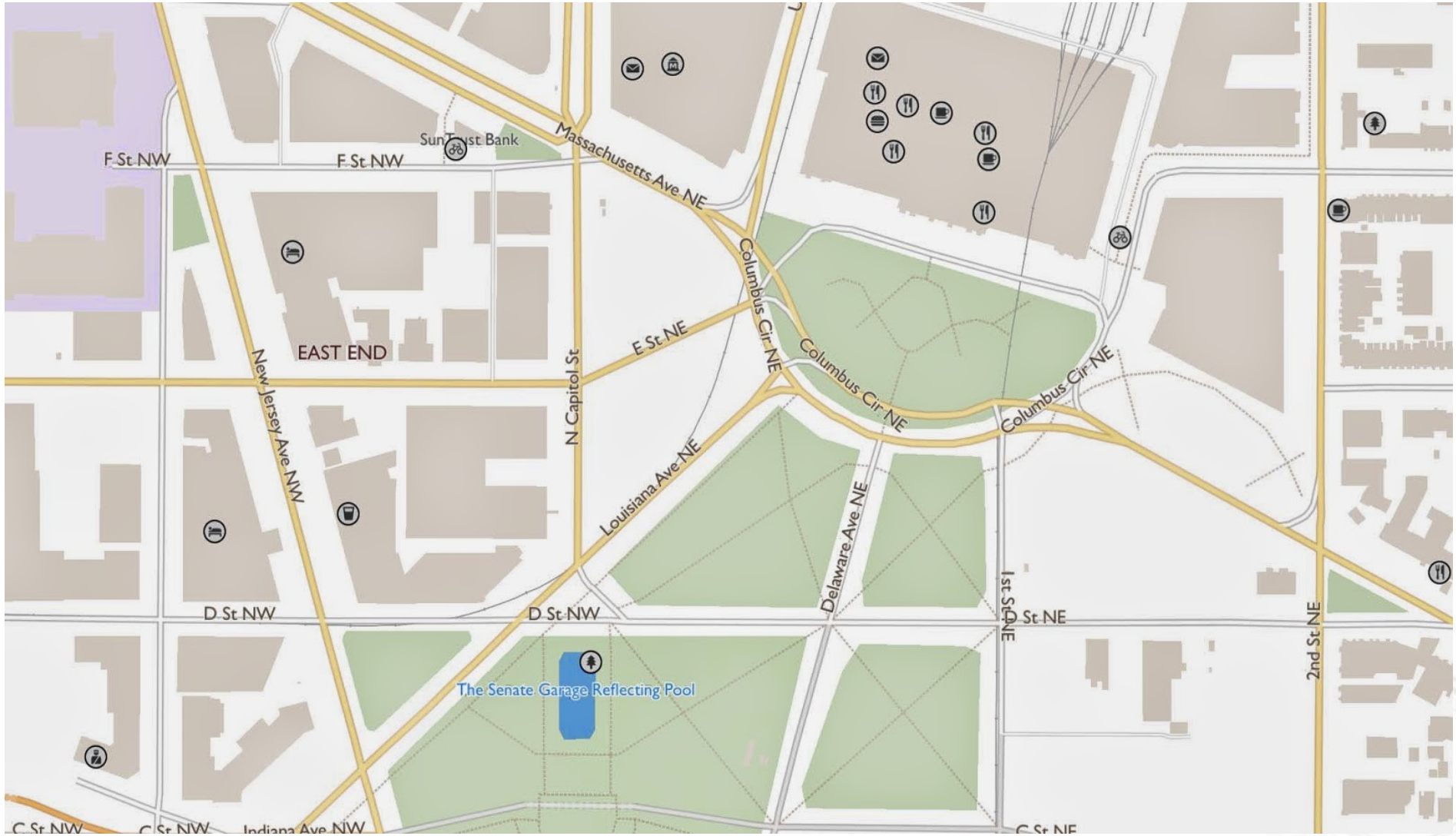
Utiliser PostGIS - avec QGIS

Interrogation + visualisation

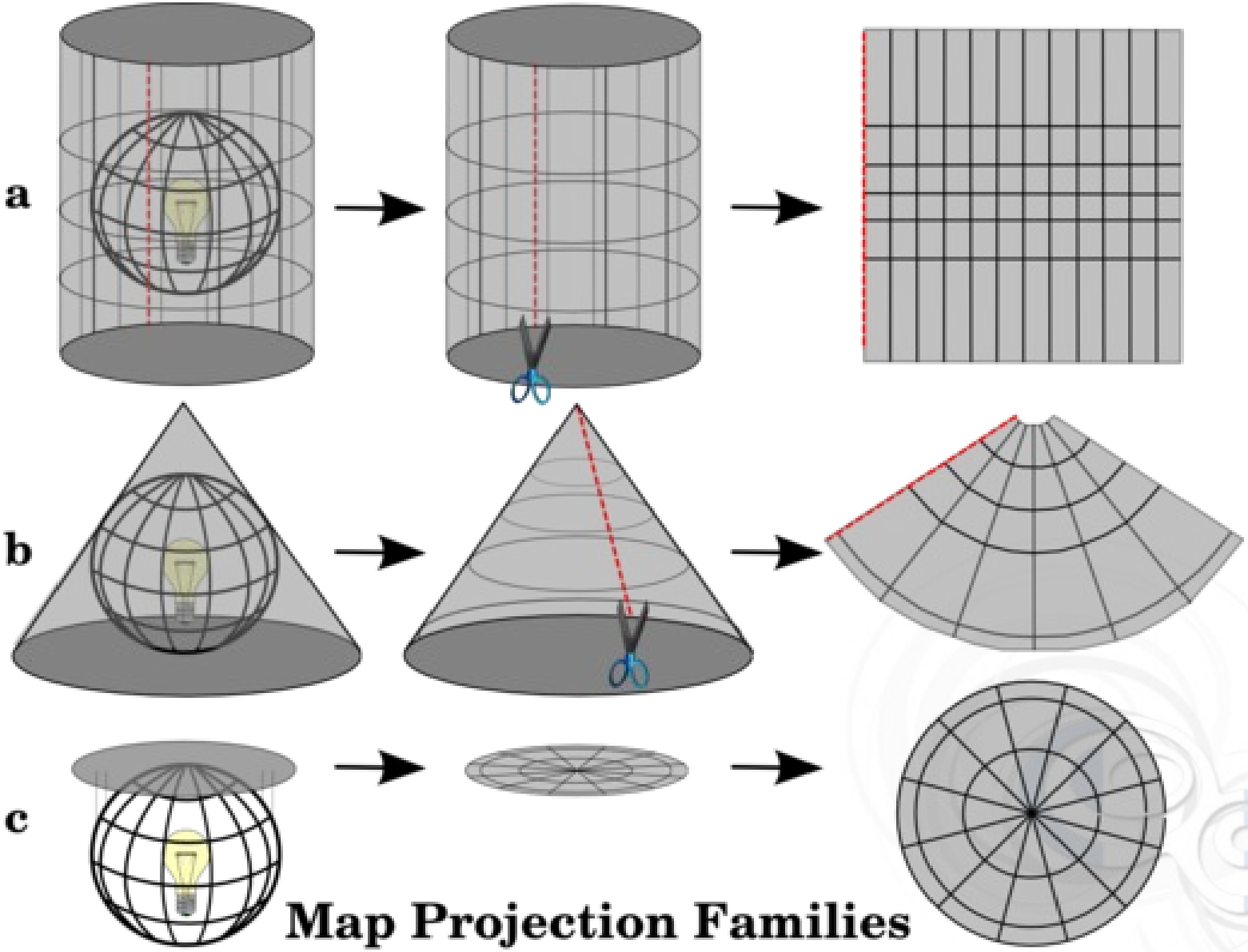


Utiliser PostGIS - Avec un serveur cartographique

Ex : Mapnik



La terre est ronde : projections



Indexation

Indexation - principe & création

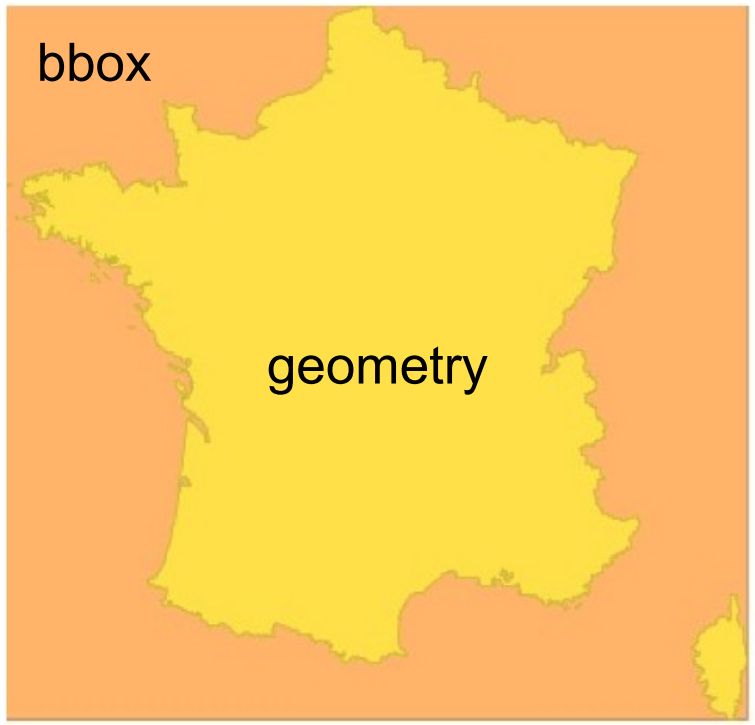
> Performances pour filtrage spatial

> Approximation des géométries par bbox

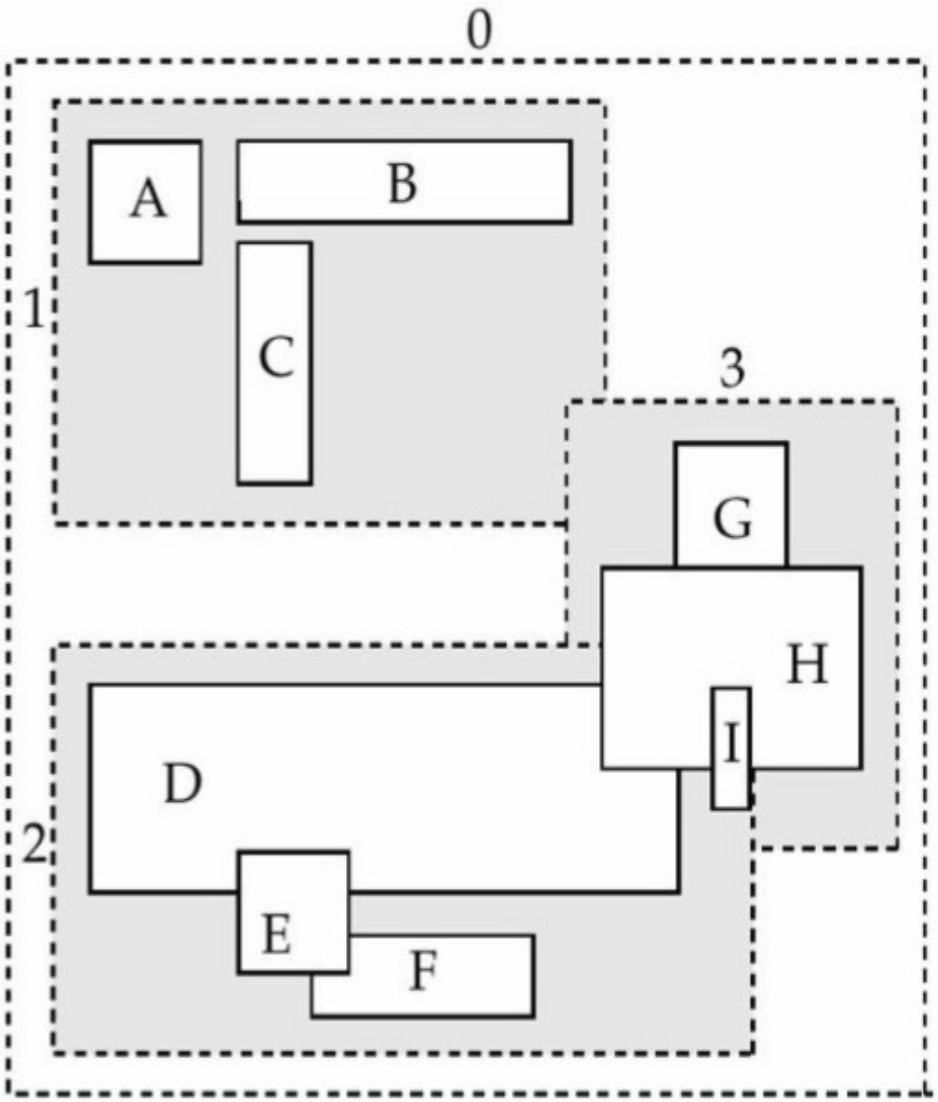
> Basé sur GIST

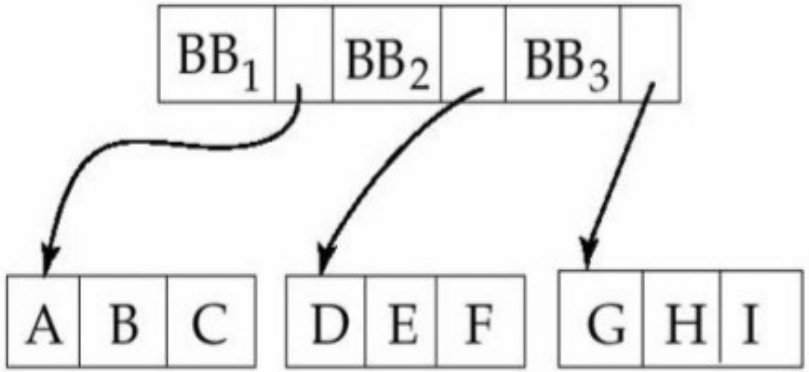
> Création index spatial :

```
CREATE INDEX index_name ON
table_name
USING GIST(geom_column_name);
```



Indexation - type R-tree

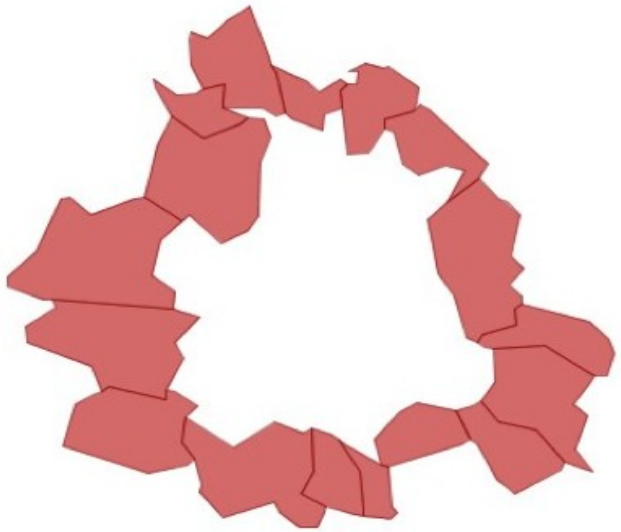




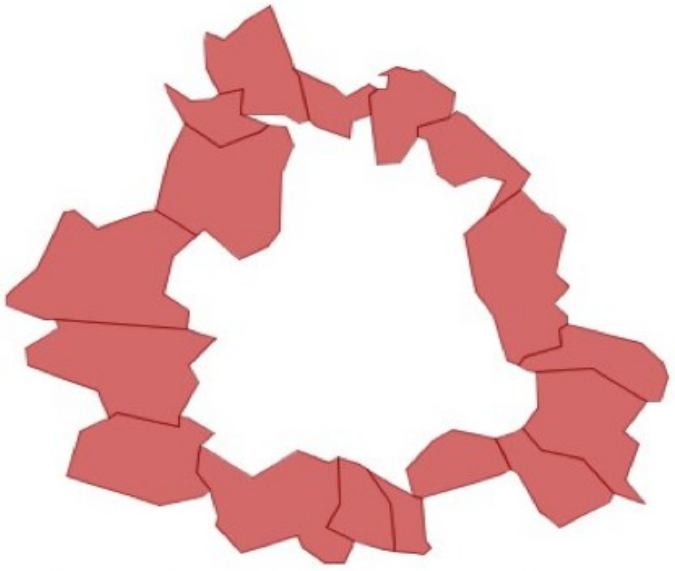
Bboxes groupées dans des régions de l'index

Indexation - exemple

```
SELECT
    c1.nom
FROM
    communes c1, communes c2
WHERE
    c2.nom = 'Toulouse'
    AND ST_Touches(c1.the_geom, c2.the_geom);
```



Sans index: temps = 150 ms



Avec index: temps = 30 ms

Use cases

Use case - distance des restaurants proches

```
select
        id, st_astext(geom) as t,
        st_distance(
                geom
                , st_transform(
                        st_setsrid(
                                st_makepoint(45.5236,-122.6750)
                                , 4326)
                        , 2154)
        ) as d
from
        restaurants;
```

Use case - restaurants à moins de 100m

```
select
    st_distance(pt, 'SRID=900913;POINT(536831 5741994)') as distance,
    name,
    pt
from
        -- beaucoup de points
        planet_osm_point
where
        amenity = 'restaurant'
        and name is not null
        -- les géométries sont à moins de 100m
        and st_dwithin(pt, 'SRID=900913;POINT(536831 5741994)', 100)
```

ST_DWITHIN => index GIST => Much Wow !

Use case - restaurants dans une commune

Principe de jointure spatiale

```
select
      r.name,
      c.name,
      r.geom,
from
            restaurant r
join
            communes c
on
            st_within(r.geom, c.geom)
```

Use case - aggrégation de géométries

Les communes de France

Les communes de France
fusionnées par département

```
SELECT ST_Union(the_geom)
FROM commune
GROUP BY code_dept;
```

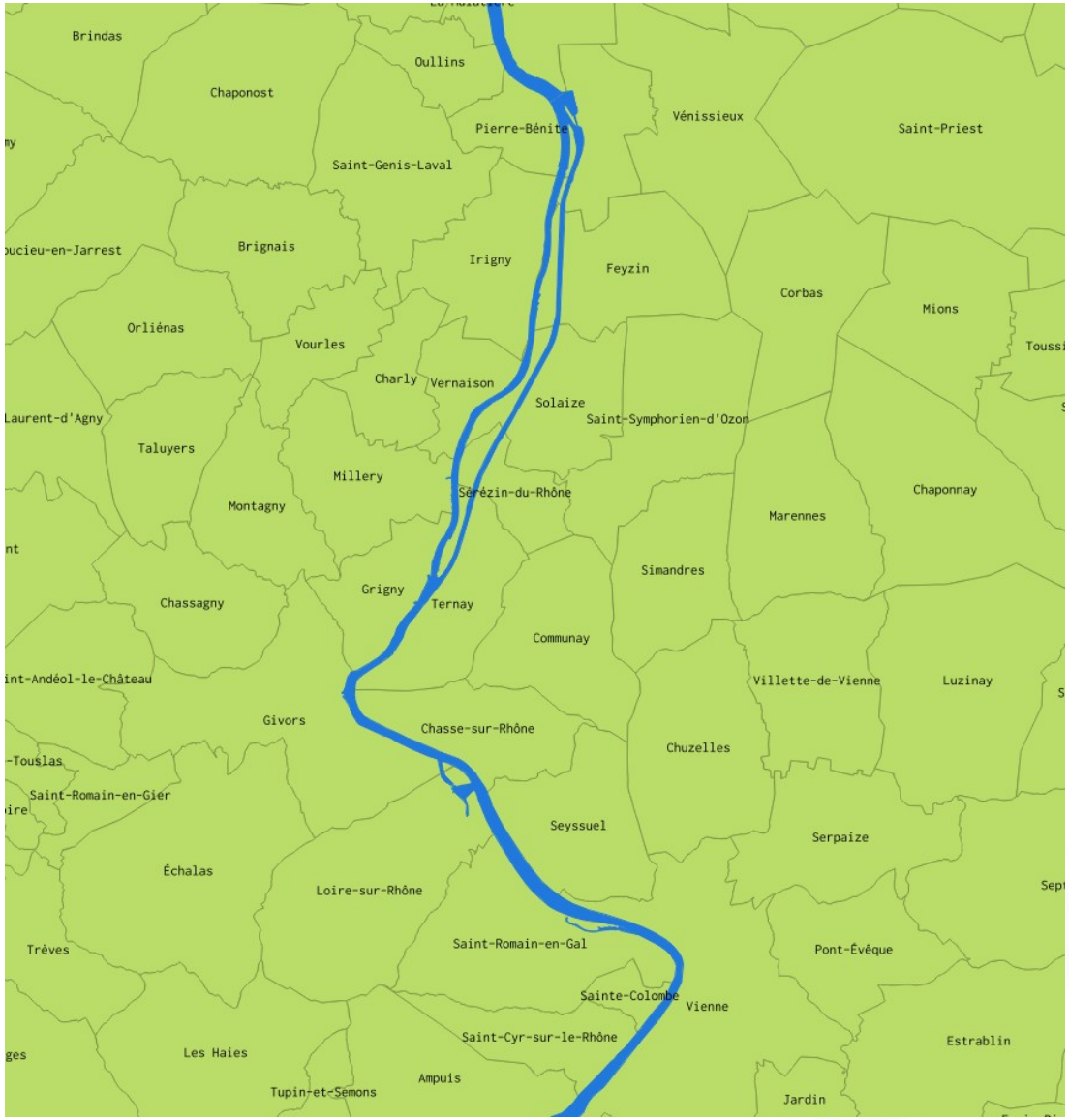
Use case - Communes le long du Rhône ?

Données OSM Rhône-Alpes

```
select
        *
from
        planet_osm_polygon
where
        admin_level = '8';
```

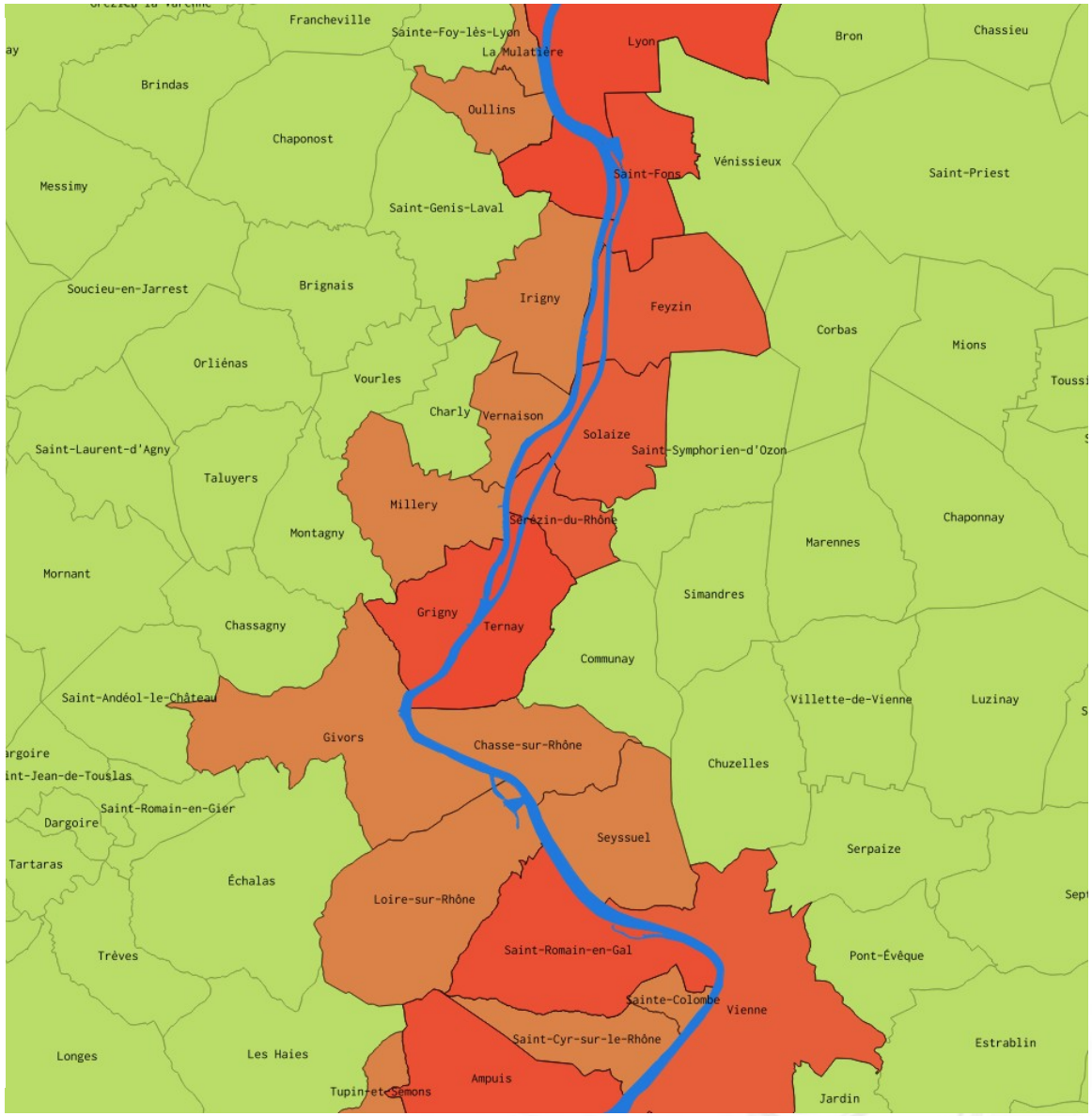
Use case - Communes le long du Rhône ?

```
select
        *
from
        planet_osm_polygon
where
        waterway = 'riverbank'
        and name = 'Le Rhône';
```



Use case - Communes le long du Rhône ?

```
WITH rhone AS (
        SELECT
                st_boundary(way) AS geom
        FROM
                planet_osm_polygon
        WHERE
                waterway = 'riverbank'
        AND
                name = 'Le Rhône'
)
SELECT
        row_number() over () as id,
        c.name,
        c.way as geom
FROM
        planet_osm_polygon AS c
JOIN
        rhone AS r
ON
        ST_Intersects(c.way, r.geom)
WHERE
        c.admin_level = '8';
```



Frameworks

Frameworks - Django + Geodjango

> django.contrib.gis

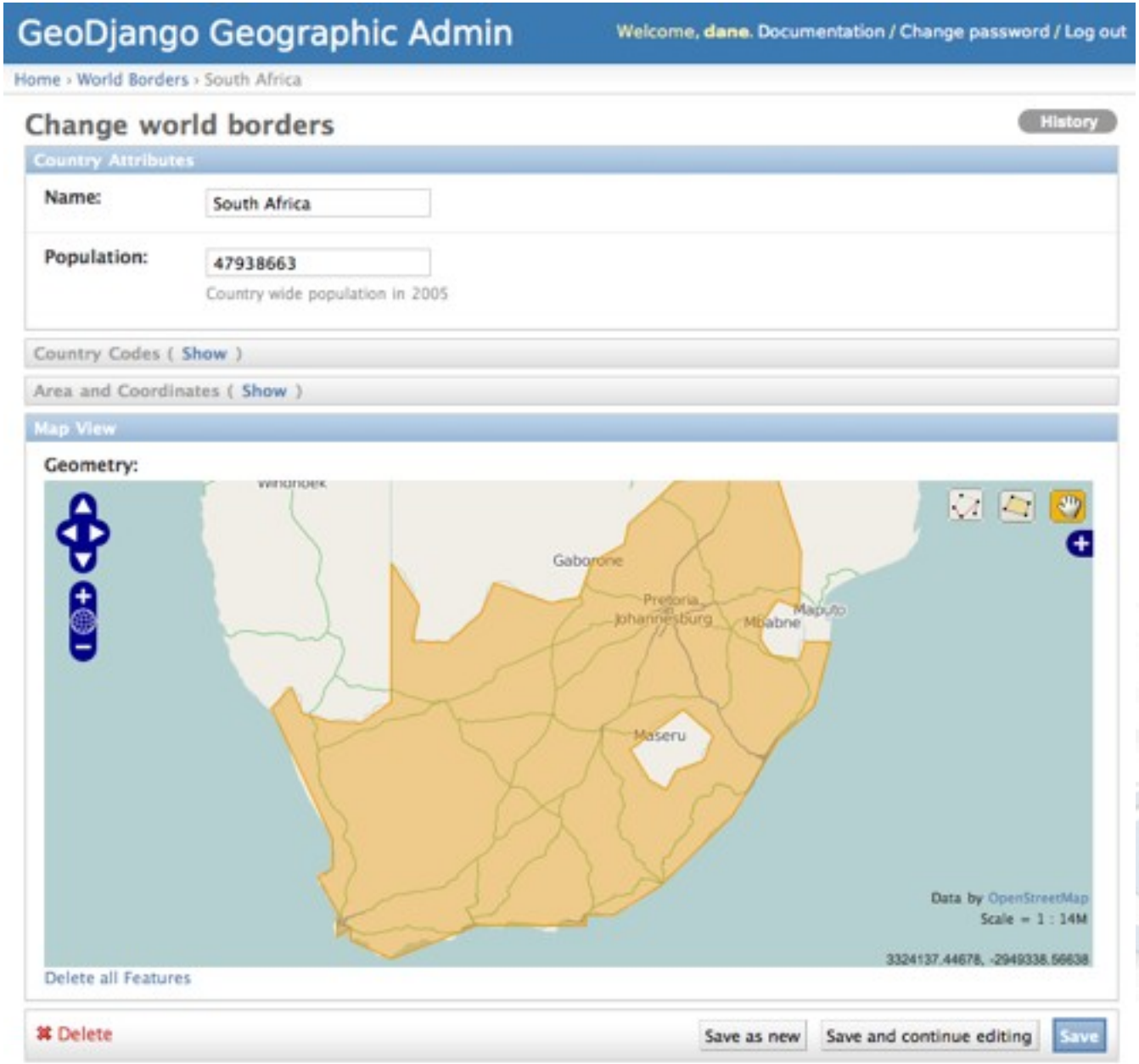
> Fonctionne avec PostGIS / Spatialite de base

> ORM avec filtres spatiaux

> Fonctionnalités Géo intégrées dans Django

> Types géographiques + widgets

Frameworks - Django + Geodjango



Plus loin que la 2D

-

La 3D

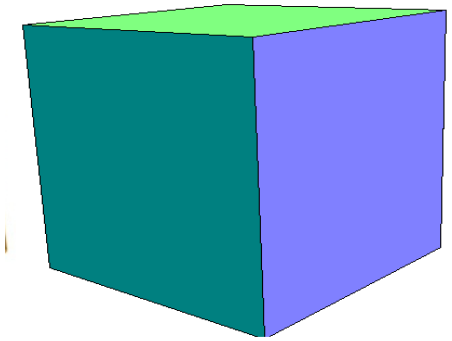
PostGIS 3D

> 3D «réelle» dans PostGIS

> Standards ISO et OGC

ISO 19125, SQL/MM, SFS 1.2.0

> Nouveaux types et fonctions



Postgis 3D – Fonction SFCGAL

SFCGAL

=

CGAL

+

CERTIFIED OGC® COMPLIANT

ISO 19107:2013

ISO 19125:2013

CGAL

Postgis 3D - Fonction SFCGAL

ST_3DIntersection

ST_Tesselate

ST_3DArea

ST_Extrude

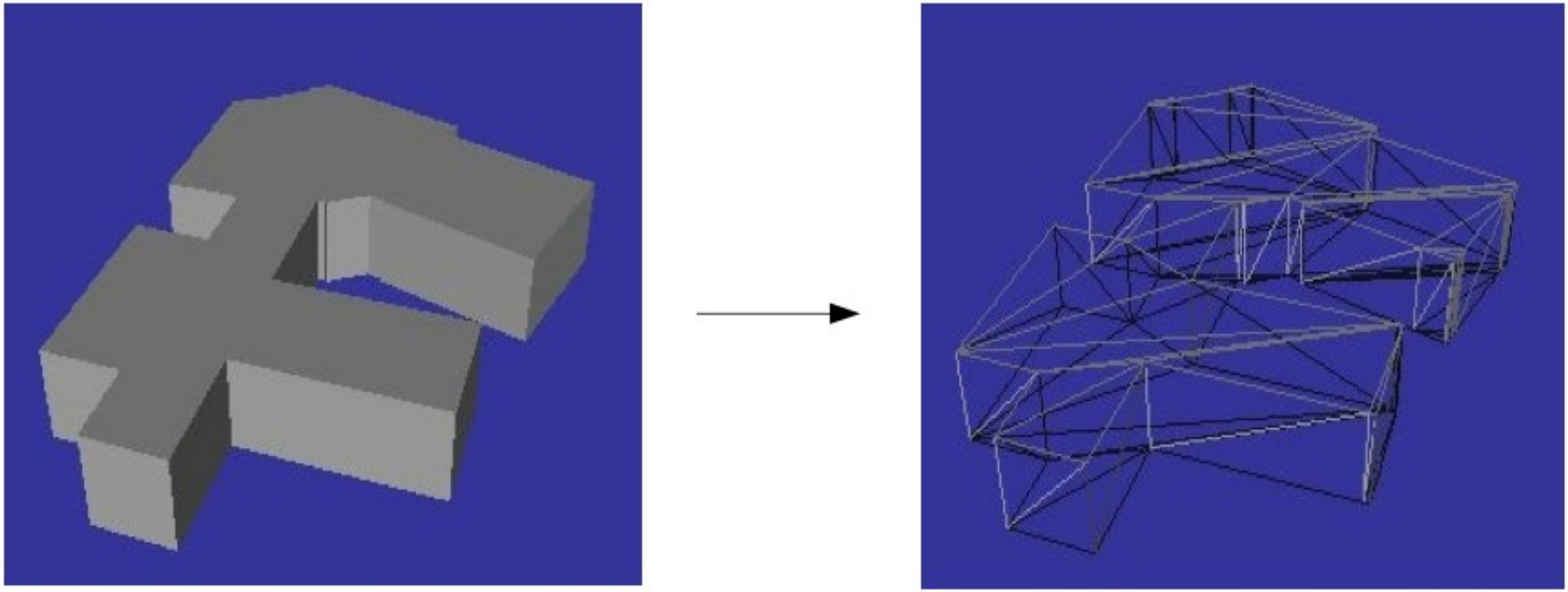
ST_ForceLHR

ST_Orientation

ST_MinkowskiSum

ST_StraightSkeleton

ST_Tesselate



ST_StraightSkeleton

2D Building Footprint

Straight Skeleton

Extrusion & roof computation

Plus loin que la 2D

-

Topologie

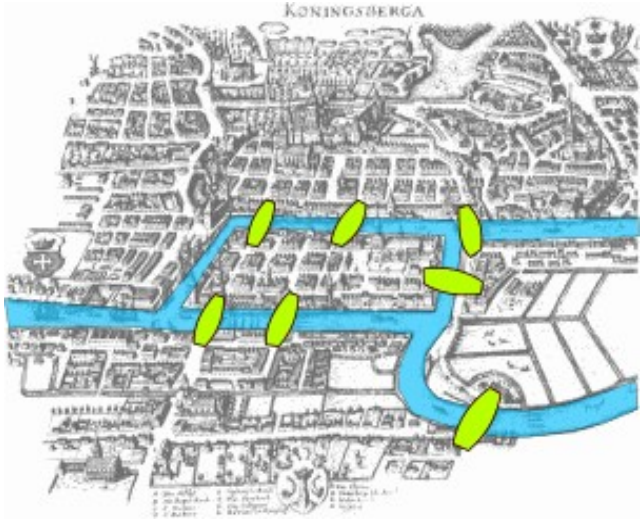
Topologie

> Relations explicites entre objets

> Représentation en graphe

> Différents modèles

Node / edge / face ↔ Noeud / Arc / Face



Topologie Postgis

> Type de donnée TopoGeometry

> Utilisation de schémas

«topology» pour les fonctions (et autres)

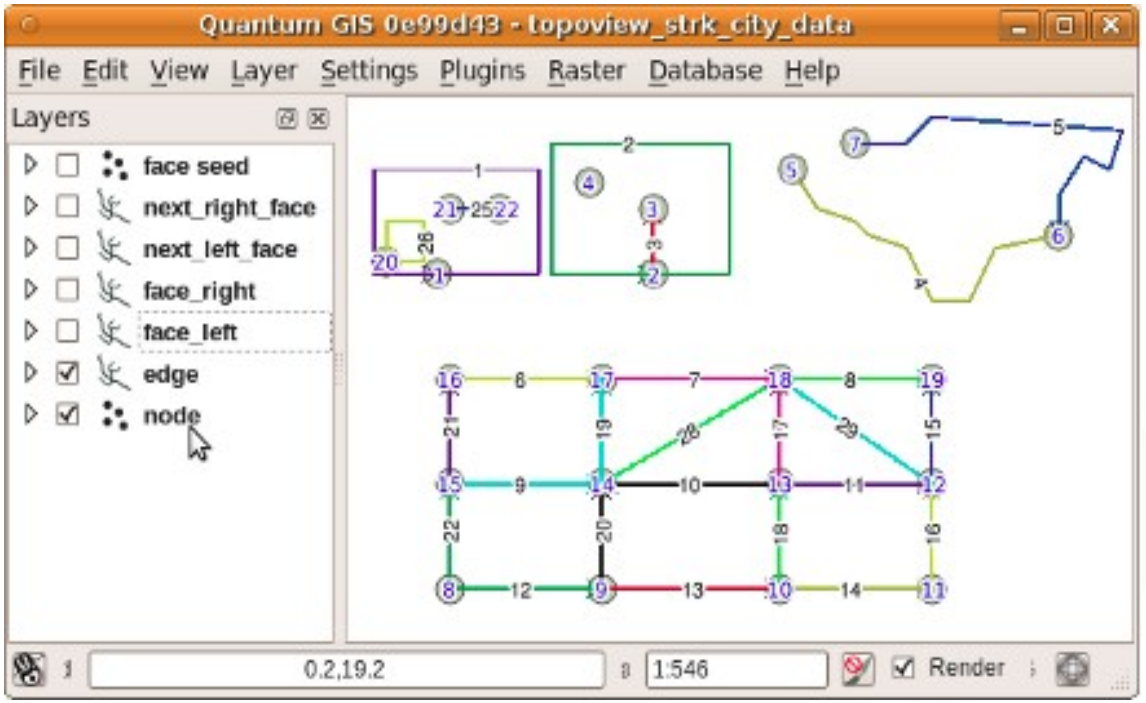
chaque topologie dans son schema

> Support complet de SQL/MM

> Intégré dans la 2.0

Sandro Santilli

Région Toscane

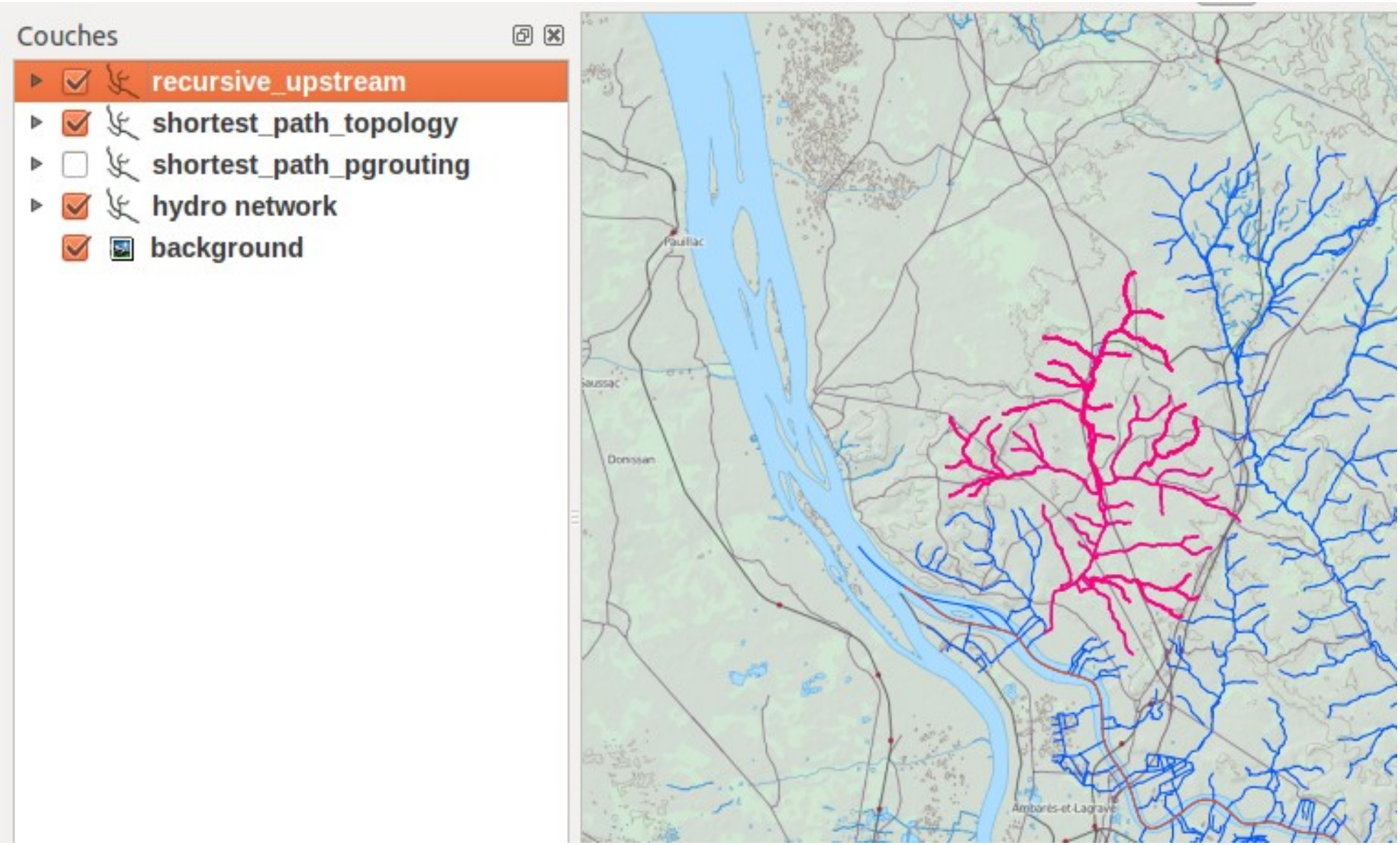


```
create table
        rec_res2 as
with recursive
        search_graph(edge_id, start_node, depth, path, length, cycle) as (
                select
                        g.edge_id, g.start_node, 1 as depth, ARRAY[g.edge_id] as path
                        , st_length(geom) as length, false as cycle
                from
                        hydro.edge as g
                where
                        edge_id = 173832
                union all
                select
                        g.edge_id
                        , g.start_node
                        , sg.depth + 1 as depth
                        , path || g.edge_id as path
                        , sg.length + st_length(g.geom) as length
                        , g.edge_id = ANY(path) as cycle
                from
                        hydro.edge as g
                join
                        search_graph as sg
                on
                        sg.start_node = g.end_node
                where
                        not cycle
                )
```

Topologie PostGIS - SQL power !

```
select
        sg.*
        , edge.geom as geom
from
        search_graph as sg
join
        hydro.edge as edge
on
        sg.edge_id = edge.edge_id
limit 1000;
```

Topologie Postgis



Plus loin que la 2D

-

Pgrouting

Pgrouting

> OSS PostgreSQL / plugin PostGIS

> Basé sur Boost Graph

> algorithmes

- Dijkstra
- A-Star
- Shooting-Star
- Problème du voyageur de commerce (TSP)
- Isochrones

Pgrouting - exemple

- Dijkstra

```
SELECT * FROM shortest_path('
                SELECT gid as id,
                         source::integer,
                         target::integer,
                         length::double precision as cost
                        FROM ways',
                5700, 6733, false, false);
```

Pgrouting - exemple

```
SELECT gid, ST_AsText(the_geom) AS the_geom
        FROM dijkstra_sp('ways', 5700, 6733);

  gid   |                                the_geom
--------+-----------------------------------------------------------------
   5534 | MULTILINESTRING((-104.9993415 39.7423284, ... ,-104.9999815 39.7444843))
   5535 | MULTILINESTRING((-104.9999815 39.7444843, ... ,-105.0001355 39.7457581))
   5536 | MULTILINESTRING((-105.0001355 39.7457581,-105.0002133 39.7459024))
    ... | ...
  19914 | MULTILINESTRING((-104.9981408 39.7320938,-104.9981194 39.7305074))
(37 rows)
```

```
SELECT gid, ST_AsText(the_geom) AS the_geom
        FROM dijkstra_sp_delta('ways', 5700, 6733, 0.1);

  gid   |                                the_geom
--------+-----------------------------------------------------------------
   5534 | MULTILINESTRING((-104.9993415 39.7423284, ... ,-104.9999815 39.7444843))
   5535 | MULTILINESTRING((-104.9999815 39.7444843, ... ,-105.0001355 39.7457581))
   5536 | MULTILINESTRING((-105.0001355 39.7457581,-105.0002133 39.7459024))
    ... | ...
  19914 | MULTILINESTRING((-104.9981408 39.7320938,-104.9981194 39.7305074))
(37 rows)
```

Plus loin que la 2D

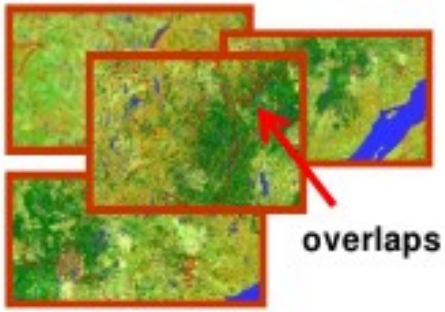
-

Les raster

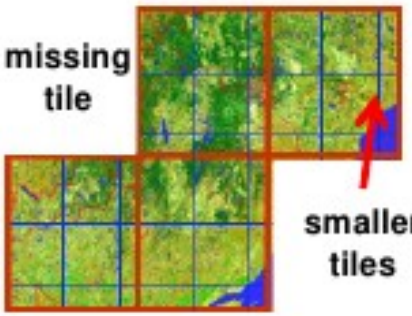
Les raster

- Analyse Raster / vecteur
- Nouveau type de données
  - ressemble à geometry
  - ... mais pour les rasters
- Multiresolution, multibande, tuiles
- Import/export (GDAL)
- Fonctions
  - statistiques, reprojection, édition, calculs
  - fonctions Vecteur/raster

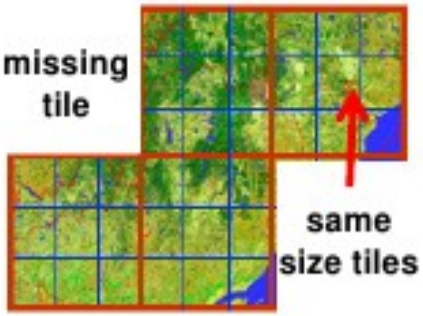
Les raster



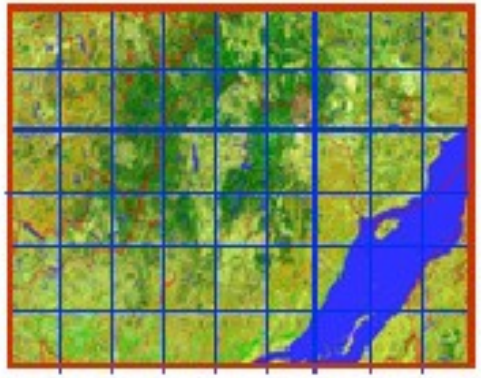
a) warehouse of untiled and unrelated images (4 images)



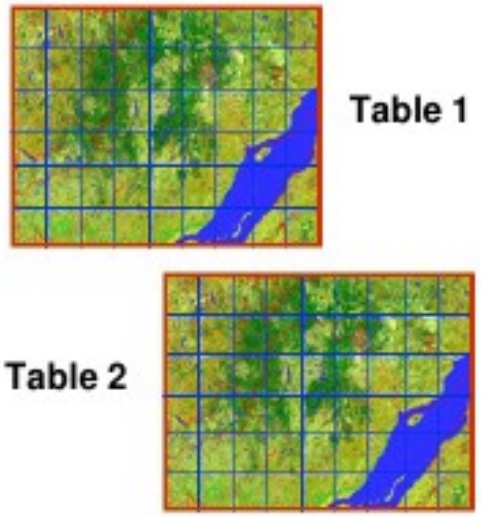
b) irregularly tiled raster coverage (36 tiles)



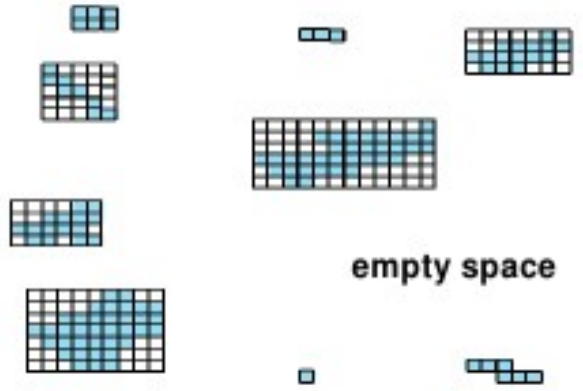
c) regularly tiled raster coverage (36 tiles)



d) rectangular regularly tiled raster coverage (54 tiles)



e) tiled images (2 tables of 54 tiles)

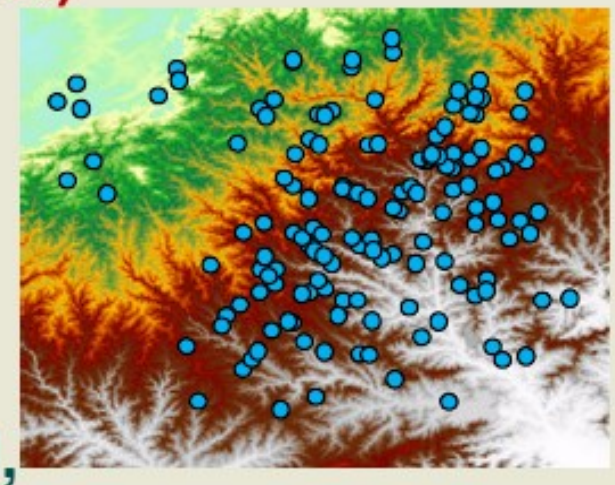


f) rasterized geometries coverage (9 lines in the table)

Les raster

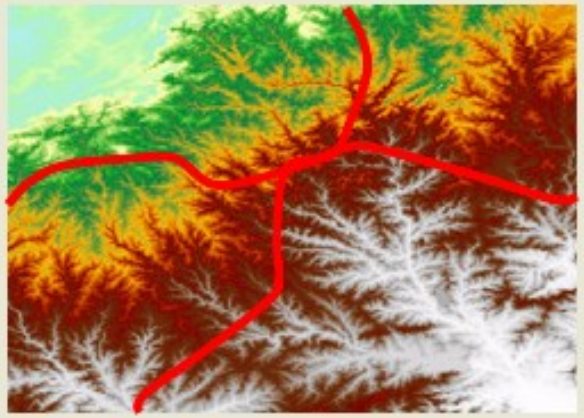
Extract ground elevation values for lidar points…

- 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)



Plus loin que la 2D

-

Nuage de points

Nuages de points

> Pointcloud Extension

Extension postgres pour stocker efficacement les nuages de points (cartographie LIDAR)

Par Paul Ramsey !

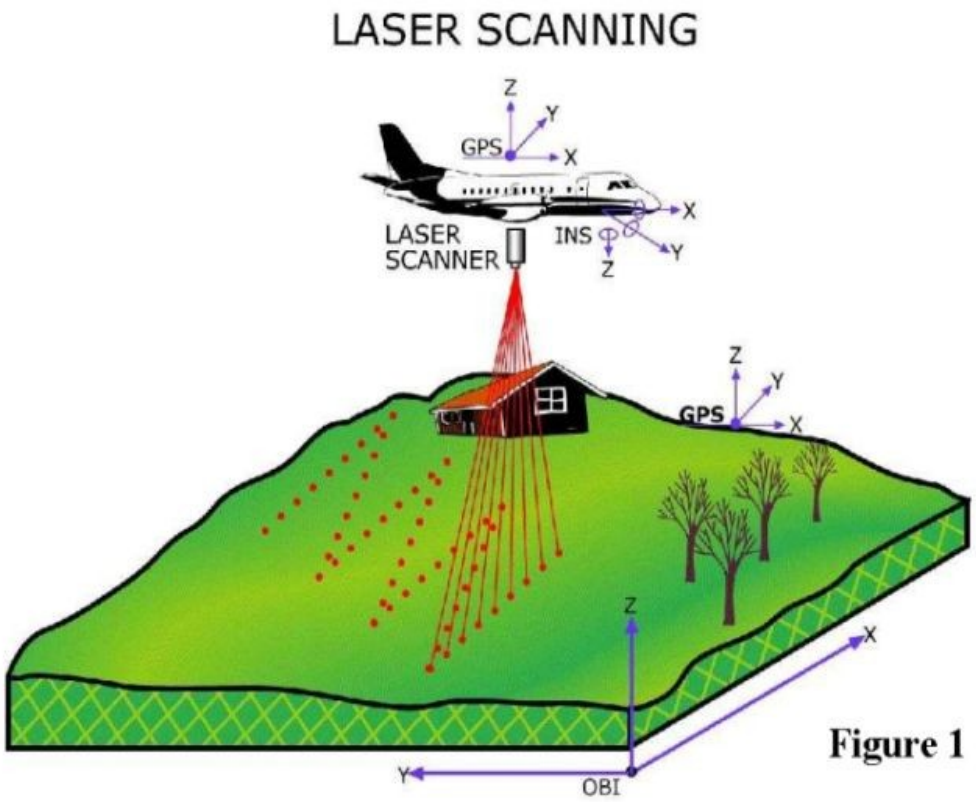
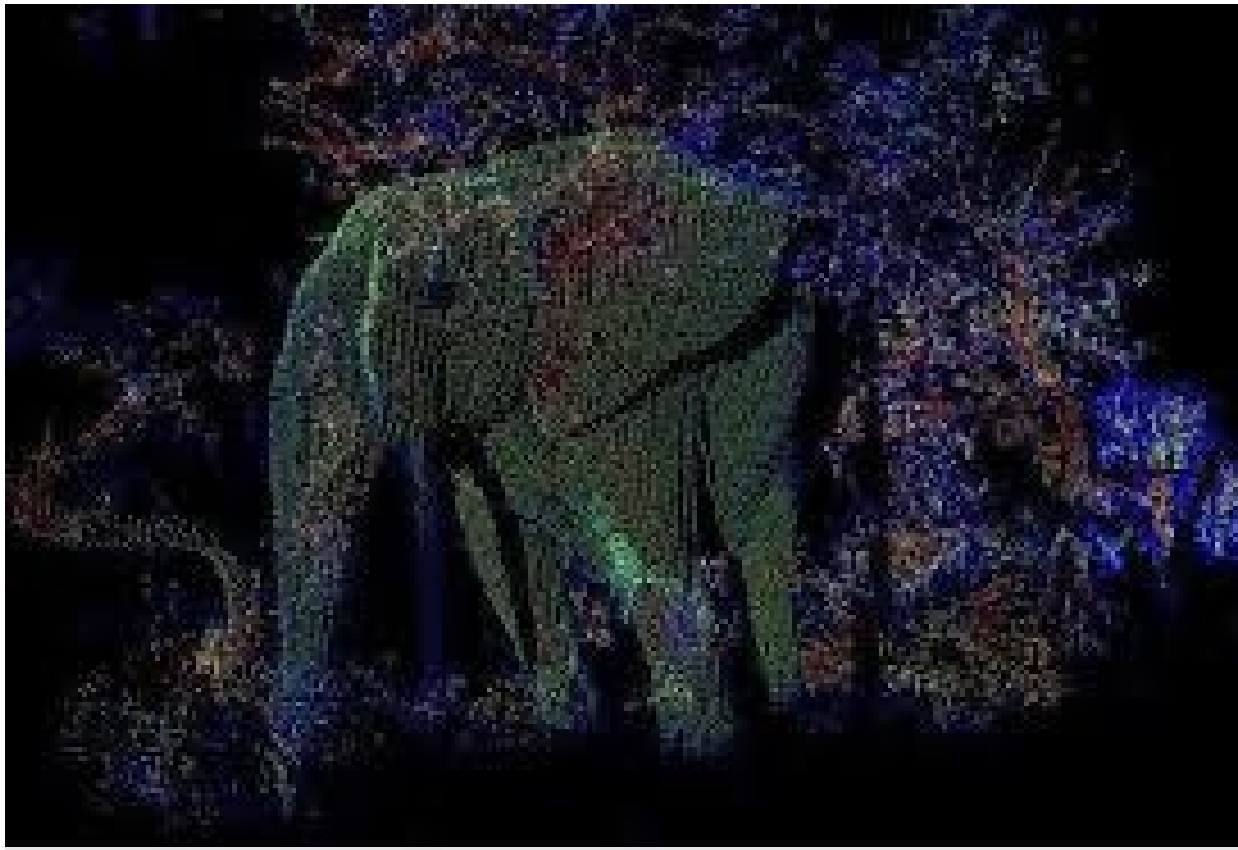


Figure 1



Des questions ?

vincent.picavet@oslandia.com

@vpicavet

https://github.com/Oslandia/presentations

http://www.oslandia.com