



Topology in PostgreSQL / PostGIS

Vincent Picavet – Oslandia

Summary

Topology ??

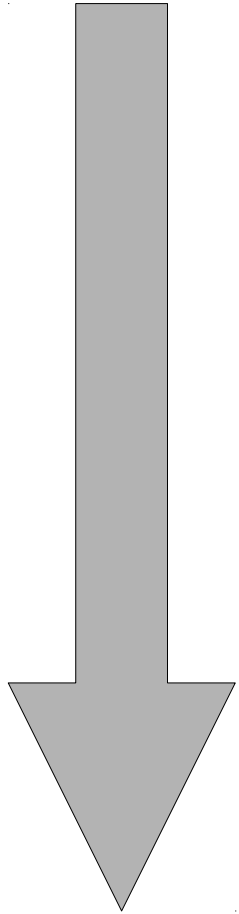
Why ?

PostGIS topology

PgRouting

Recursive queries

Example on Hydrology network



Topology ???



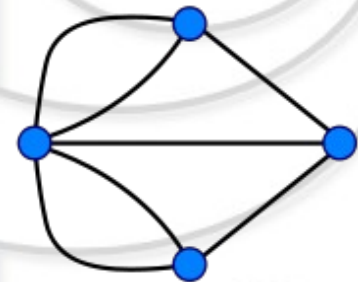
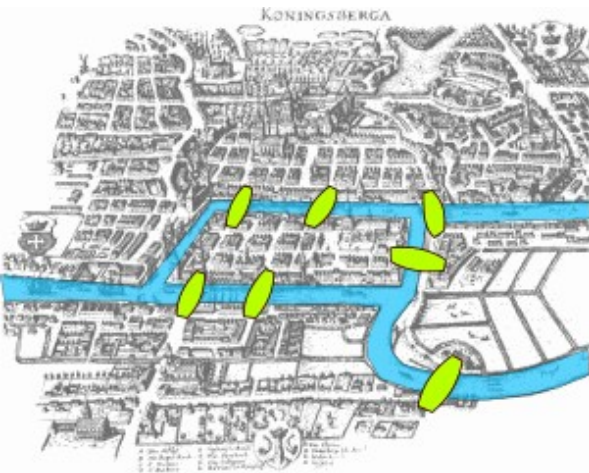
Spaghetti model



Beware of the spaghetti monster !

Topology - Graphs

- Explicit relations between objects
- Graph representation
- Various types of graphs and networks
 - Node / edge / face



Graphs in database



Edi Data - s92s08260 (s92s08260.5432) - TEST_PROD_08_07 - nav_noeud_app

Fichier Édition Affichage Aide

100 lignes.

	id [PK] bigint	id_nav bigint	insee character var	the_geom geometry
1	1	13297	29019	01010000208E6800001801
2	2	13429	29019	01010000208E68000063A1A
3	3	13481	29019	01010000208E680000091A
4	4	13525	29019	01010000208E680000E850E
5	5	13539	29019	01010000208E680000700D
6	6	13609	29019	01010000208E68000060AD0
7	7	13629	29019	01010000208E680000C6126
8	8	13682	29019	01010000208E680000D8B89
9	9	13697	29019	01010000208E68000086617
10	10	13704	29019	01010000208E680000338CF
11	11	13708	29019	01010000208E68000035881
12	12	13711	29019	01010000208E680000D2833
13	13	13719	29019	01010000208E68000014216
14	14	13721	29019	01010000208E680000D552C

Facteur remplissage

Edit Data - s92s08260 (s92s08260.5432) - TEST_PROD_08_07 - nav_troncon_app

Fichier Édition Affichage Aide

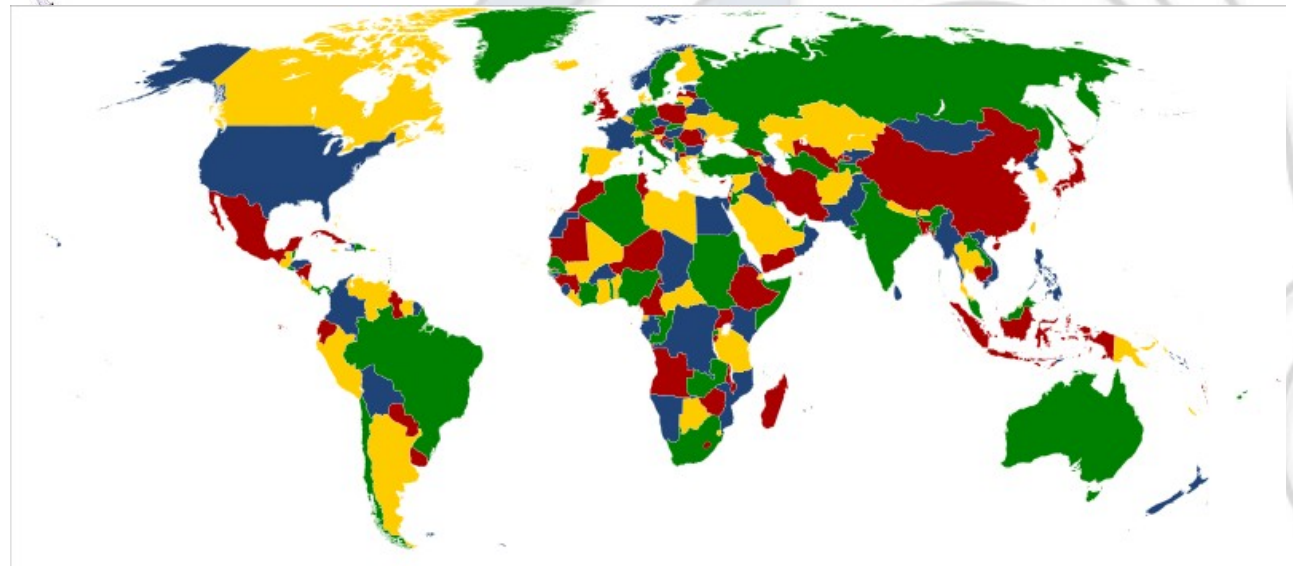
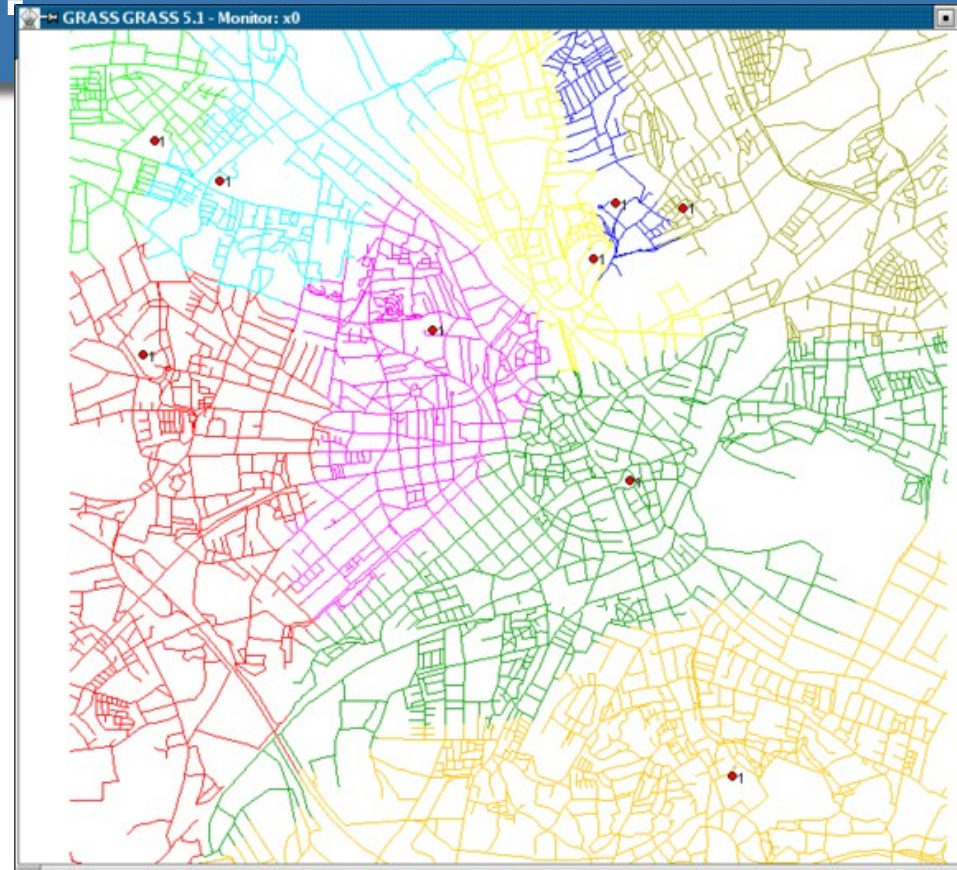
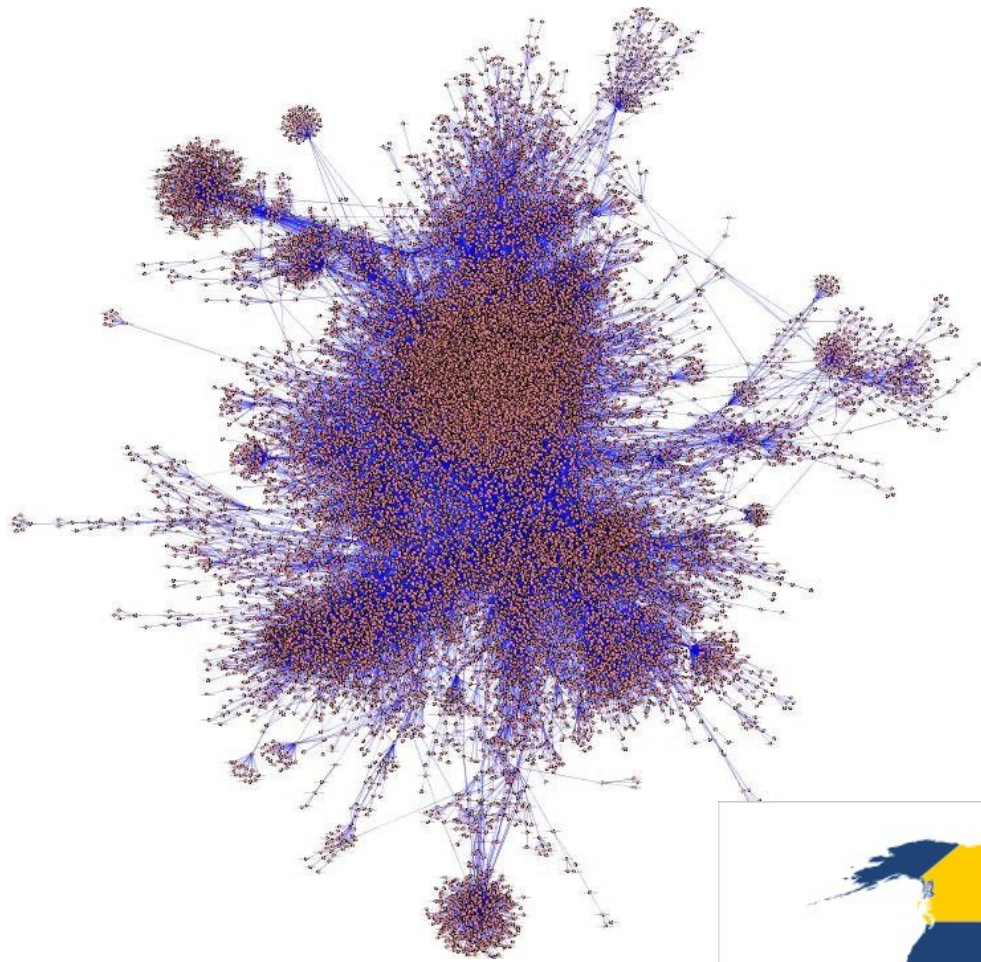
100 lignes.

	id [PK] bigint	id_nav bigint	id_noeud_fro bigint	id_noeud_to bigint	id_voie bigint	insee character var	longueur integer
1	1	95886	74	77	2028	29019	128
2	2	95905	464	473	2715	29019	62
3	3	95910	435	430	2130	29019	33
4	4	95922	967	981	2001	29019	43
5	5	95974	3455	3416	1905	29019	58
6	6	95980	1039	1040	2740	29019	15
7	7	95982	372	337	3477	29019	62
8	8	95988	103	104	781	29019	15
9	9	96000	1013	1054	3000	29019	45
10	10	96009	428	400	3447	29019	107
11	11	96010	843	790	4341	29019	47
12	12	96015	1673	1663	3160	29019	63

Why ?



Why ?



Why ?

- Normalized spatial data
- Standard interface
- Topological integrity
- Reduced storage size
- Explicit spatial relationships

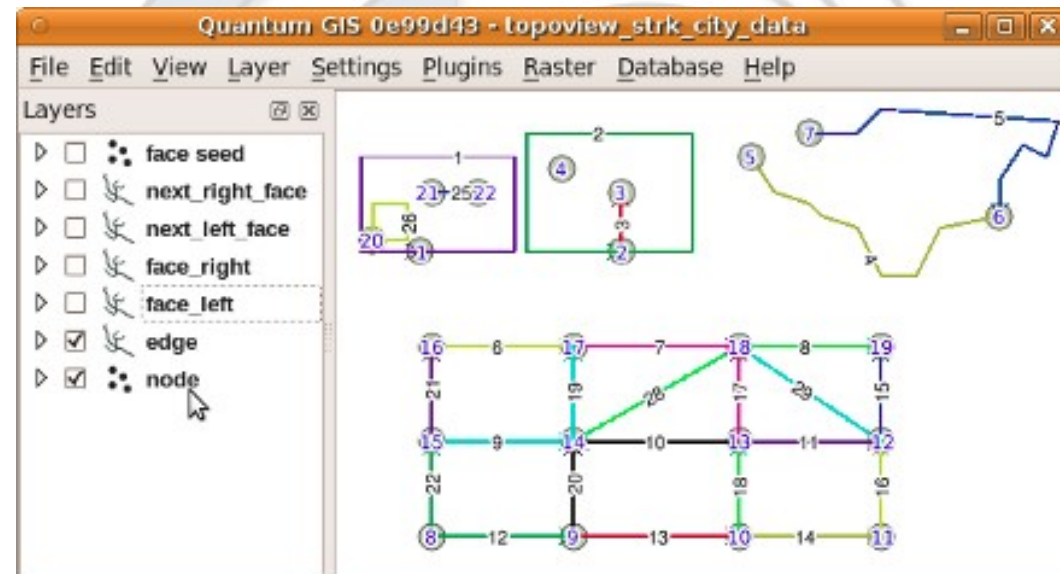


PostGIS Topology



PostGIS Topology

- Initiated long ago
- ISO SQL/MM implementation
- In PostGIS 2.0
- Sandro Santilli (Toscana Region, IT)
- Still under development



PostGIS Topology – Concepts

- Node / Edge / Face model
- Uses schema :
 - 1 General «topology» schema
 - 1 schema per topology
 - mytopo.{edge, face, node, relation}
- Metadata tables on topologies and layers
 - topology.{topology, layer}
- TopoGeometry Datatype
- Cast to geometry

PgRouting



PgRouting

- OSS PostgreSQL / PostGIS plugin
- Based on Boost Graph
- Live computation inside the database
- Various algorithms
 - Shortest Path Dijkstra
 - Shortest Path A-Star
 - Shortest Path Shooting-Star
 - Traveling Salesperson Problem (TSP)
 - Driving Distance calculation (Isolines)



PgRouting – data

- For Dijkstra : edges
- gid, source, target, cost
- Source and target : node ids
- Cost : any value
 - Field value
 - pre-computed (length)
 - computed on the fly
 - computed with complex queries on live data

PgRouting – Example

■ Dijkstra

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

```
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))



Recursive Queries

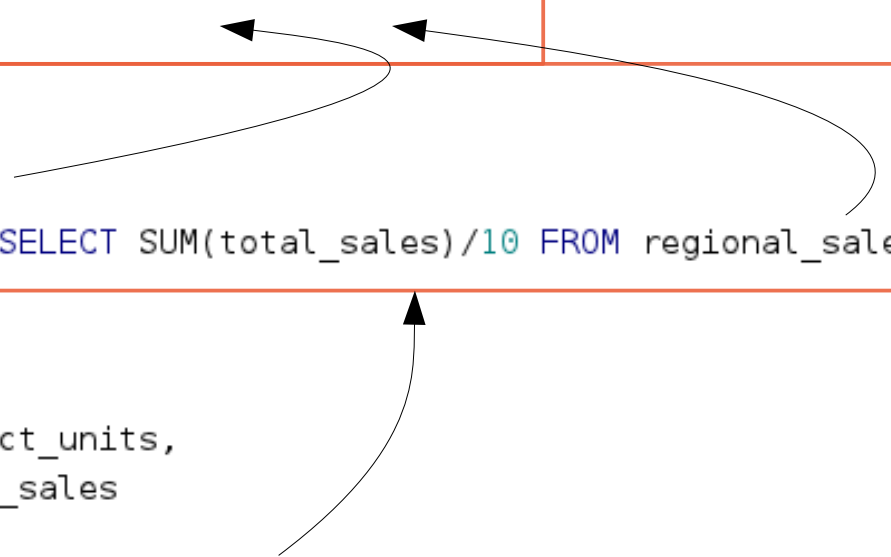


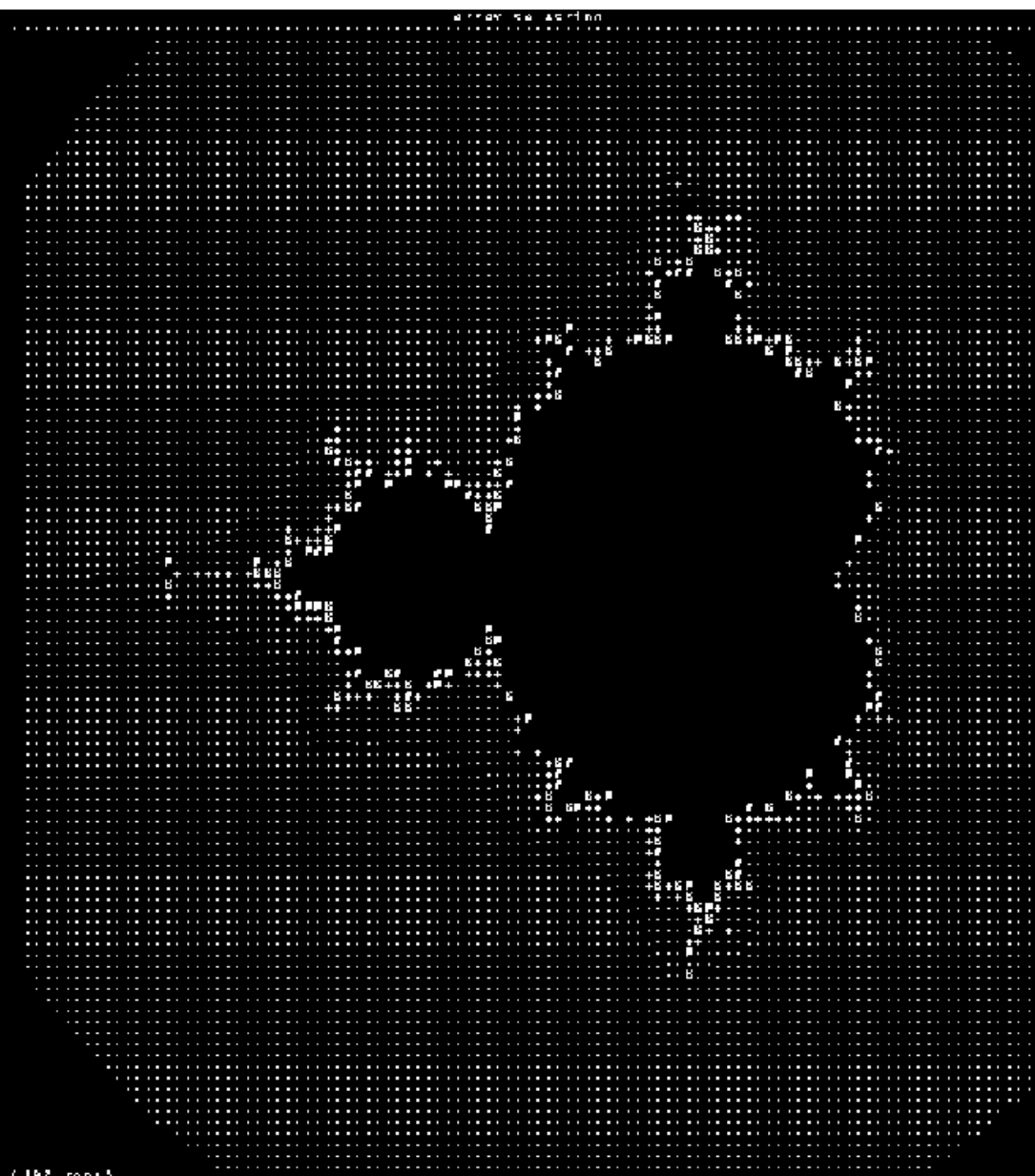
Recursive queries – principles

- Common Table Expressions
- CTE in PostgreSQL 8.4+
- ~ = temporary table
- WITH RECURSIVE option

```
WITH RECURSIVE t(n) AS (  
    VALUES (1)  
    UNION ALL  
    SELECT n+1 FROM t WHERE n < 100  
)  
SELECT sum(n) FROM t;
```

```
WITH regional_sales AS (  
    SELECT region, SUM(amount) AS total_sales  
    FROM orders  
    GROUP BY region  
) , top_regions AS (  
    SELECT region  
    FROM regional_sales  
    WHERE total_sales > (SELECT SUM(total_sales)/10 FROM regional_sales)  
)  
SELECT region,  
       product,  
       SUM(quantity) AS product_units,  
       SUM(amount) AS product_sales  
FROM orders  
WHERE region IN (SELECT region FROM top_regions)  
GROUP BY region, product;
```





```

WITH RECURSIVE
x(i)
AS (
    VALUES(0)
UNION ALL
    SELECT i + 1 FROM x WHERE i < 101
),
Z(Ix, Iy, Cx, Cy, X, Y, I)
AS (
    SELECT Ix, Iy, X::float, Y::float, X::float, Y::float, 0
    FROM
        (SELECT -2.2 + 0.031 * i, i FROM x) AS xgen(x,ix)
    CROSS JOIN
        (SELECT -1.5 + 0.031 * i, i FROM x) AS ygen(y,iy)
    UNION ALL
    SELECT Ix, Iy, Cx, Cy, X * X - Y * Y + Cx AS X, Y * X * 2 + Cy, I + 1
    FROM Z
    WHERE X * X + Y * Y < 16.0
    AND I < 27
),
Zt (Ix, Iy, I) AS (
    SELECT Ix, Iy, MAX(I) AS I
    FROM Z
    GROUP BY Iy, Ix
    ORDER BY Iy, Ix
)
SELECT array_to_string(
    array_agg(
        SUBSTRING(
            ' .,.,,-----++++%:%:%@#@# ' ,
            GREATEST(I,1),
            1
        )
    ), ''
)
FROM Zt
GROUP BY Iy
ORDER BY Iy;

```


Let's eat SQL !

Hydrology network



Our data

Table name : **tr**



Fichier Éditer Vue Couche Préférences Extension Vecteur Base de donnée Raster Aide

Couches

- ☐ recursive_upstream_topo
- ☐ recursive_upstream
- ☐ shortest_path_topology
- ☐ shortest_path_pgrouting
- ☒ hydro network
- ☒ background

Attribute table - hydro network :: 0 / 18936 feature(s) selected

	gid	source	target	hname	cost
0	17681	3042	3041	ruisseau de...	13.1468627...
1	50006	4363	4376	ruisseau de...	154.831357...
2	107308	4427	4443	ruisseau la ...	70.4784694...
3	110767	4810	4816	ruisseau le ...	426.452159...
4	8923	4892	4827	ruisseau de...	1648.21133...
5	109594	5158	5264	rivière la di...	946.014083...
6	45039	5407	5429	NULL	114.028638...
7	105937	5480	5594	ruisseau le ...	824.626701...
8	104620	5481	5518	ruisseau la ...	243.004034...

☒ Contrôle de l'ordre de rendu des couches

Topology

- Geometry table = spaghetti
- Custom attribute-based topology :
 - source, target (and cost)

```
select * from tr limit 10;
```

Partie de données						
Expliquer (Explain) Messages Historique						
gid	source	target	hname	cost	geom	
integer	integer	integer	character varying(127)	double precision	geometry(MultiLin	
17681	3042	3041	ruisseau de chaize	13.146862743	01050000206A08	
50006	4363	4376	ruisseau de villeval	154.83135760	01050000206A08	
107308	4427	4443	ruisseau la méouzette	70.478469414	01050000206A08	

- Or build a PostGIS topology based on geom

```
-- Create a topology
SELECT topology.CreateTopology('hydro', 2154);
-- 1

-- we put the postgis topology features for hydro network in another table
CREATE TABLE tr_topo (gid integer);

-- Add a layer
SELECT topology.AddTopoGeometryColumn('hydro', 'public',
    'tr_topo', 'topogeom', 'MULTILINESTRING');
-- 1

-- Populate the layer and the topology from tr geometry features
INSERT into tr_topo (gid, topogeom)
    SELECT gid, topology.toTopoGeom(geom, 'hydro', 1) FROM tr;
```

- [-] Schémas (3)
 - [-] **hydro**
 - [-] Collationnements (0)
 - [-] Domaines (0)
 - [-] Configurations FTS (0)
 - [-] Dictionnaires FTS (0)
 - [-] Analyseurs FTS (0)
 - [-] Modèles FTS (0)
 - [-] Fonctions (0)
 - [+] Séquences (5)
 - [-] Tables (4)
 - [+] edge_data
 - [+] face
 - [+] node
 - [+] relation
 - [-] Fonctions trigger (0)
 - [-] Types (0)
 - [-] Vues (1)
 - [+] edge

```
select * from hydro.edge limit 10;
```

neau sortie

ortie de données

Expliquer (Explain)

Messages

Historique

	edge_id integer	start_node integer	end_node integer	next_left_edge integer	next_right_edge integer	left_face integer	right_face integer	geom geometry(LineString)
1	175256	190369	190361	175230	-175243	0	0	01020000206A080
2	167356	183762	181917	166725	167356	0	0	01020000206A080

```
select * from tr_topo limit 10;
```

eau sortie

ortie de données

Expliquer (Explain)

Messages

gid integer	topogeom topology.topogeometry
116768	(1,1,163704,2)
116767	(1,1,163705,2)

An aerial photograph of a river delta system. A large, light-colored central channel flows from the bottom left towards the top right. Numerous smaller, winding tributaries branch off from this main channel, creating a complex, web-like pattern across the landscape. The land is a mix of dark green, brown, and yellowish-green, indicating different types of vegetation and possibly some agricultural areas. The overall scene is a high-angle view of a natural waterway system.

**Find our way
on the water**

PgRouting on custom topology

```
/* shortest path */  
select * from shortest_path('select gid as id, source, target, cost from tr', 15895, 20196, false, false);
```

eau sortie

rtie de données Expliquer (Explain) Messages Historique

vertex_id integer	edge_id integer	cost double precision
15895	79282	1498.6399958
15655	99961	3757.3354126
15067	22037	698.88553716

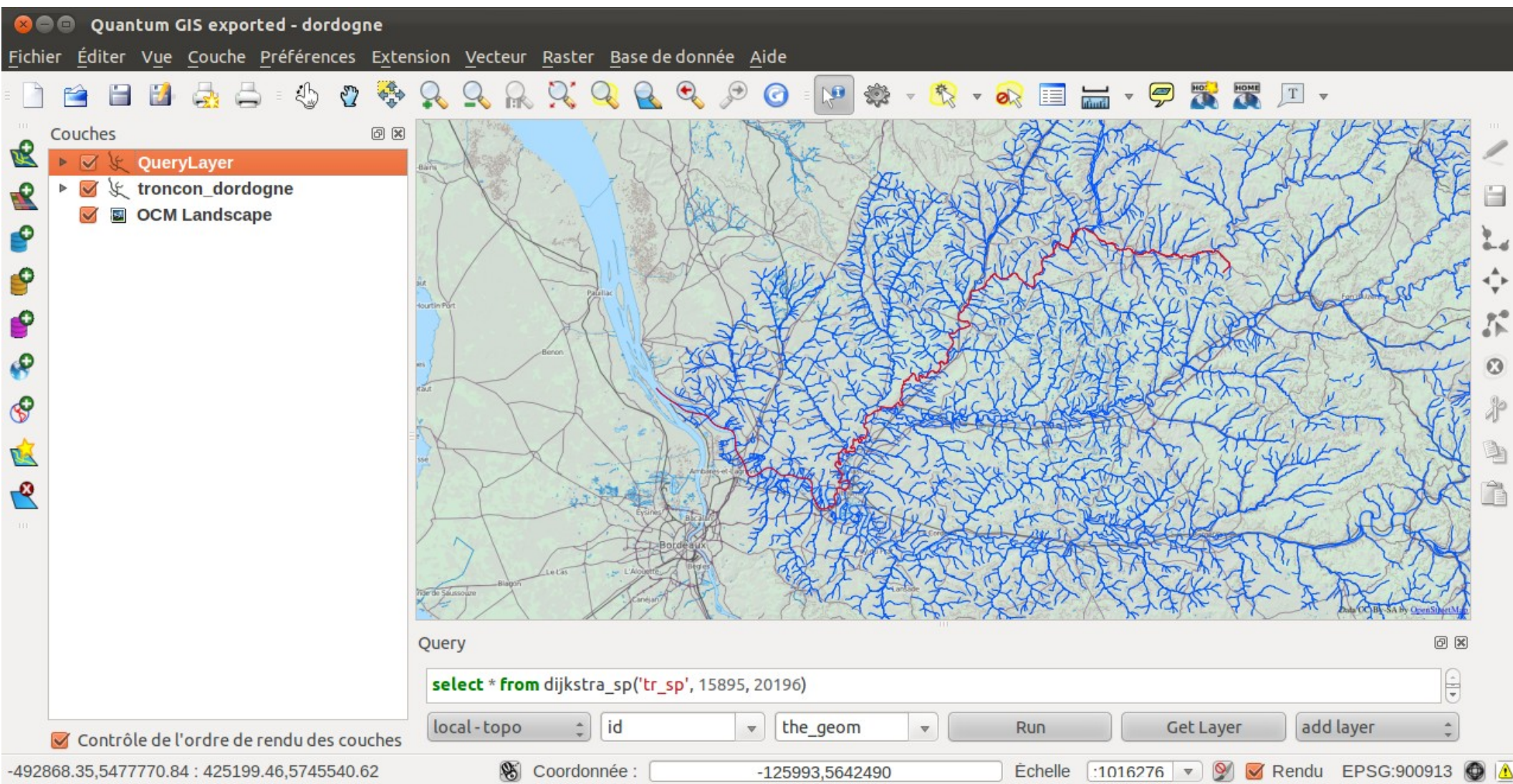
```
create or replace view tr_sp as select gid, source, target, cost as length, geom as the_geom from tr  
-- get shortest path and geometries with wrapper  
select * from dijkstra_sp('tr_sp', 15895, 20196);
```

eau sortie

rtie de données Expliquer (Explain) Messages Historique

id integer	gid integer	the_geom geometry
1	79282	01050000206A080000010000
2	99961	01050000206A080000010000
3	22037	01050000206A080000010000

PgRouting result



PgRouting on PostGIS topology

```
-- find corresponding topology edge_id and nodes for specified gid
select
    tr_topo.gid, edge_id, start_node, end_node
from
    tr_topo
join
    hydro.relation
on
    (tr_topo.topogeom).id = hydro.relation.topogeo_id
join
    hydro.edge
on
    hydro.relation.element_id = hydro.edge.edge_id
where
    tr_topo.gid in (79282, 31879);
```

au sortie

Partie de données Expliquer (Explain) Messages Historique

gid integer	edge_id integer	start_node integer	end_node integer
79282	166371	182521	177735
31879	173839	189555	189556

```
create or replace view hydroedge_sp as
    select edge_id as gid, start_node as source, end_node as target, st_length(geom) as length, geom as the_geom
    from hydro.edge;
```

```
-- get shortest path and geometries with wrapper
select * from dijkstra_sp('hydroedge_sp', 182521, 189555);
```

au sortie

Partie de données Expliquer (Explain) Messages Historique

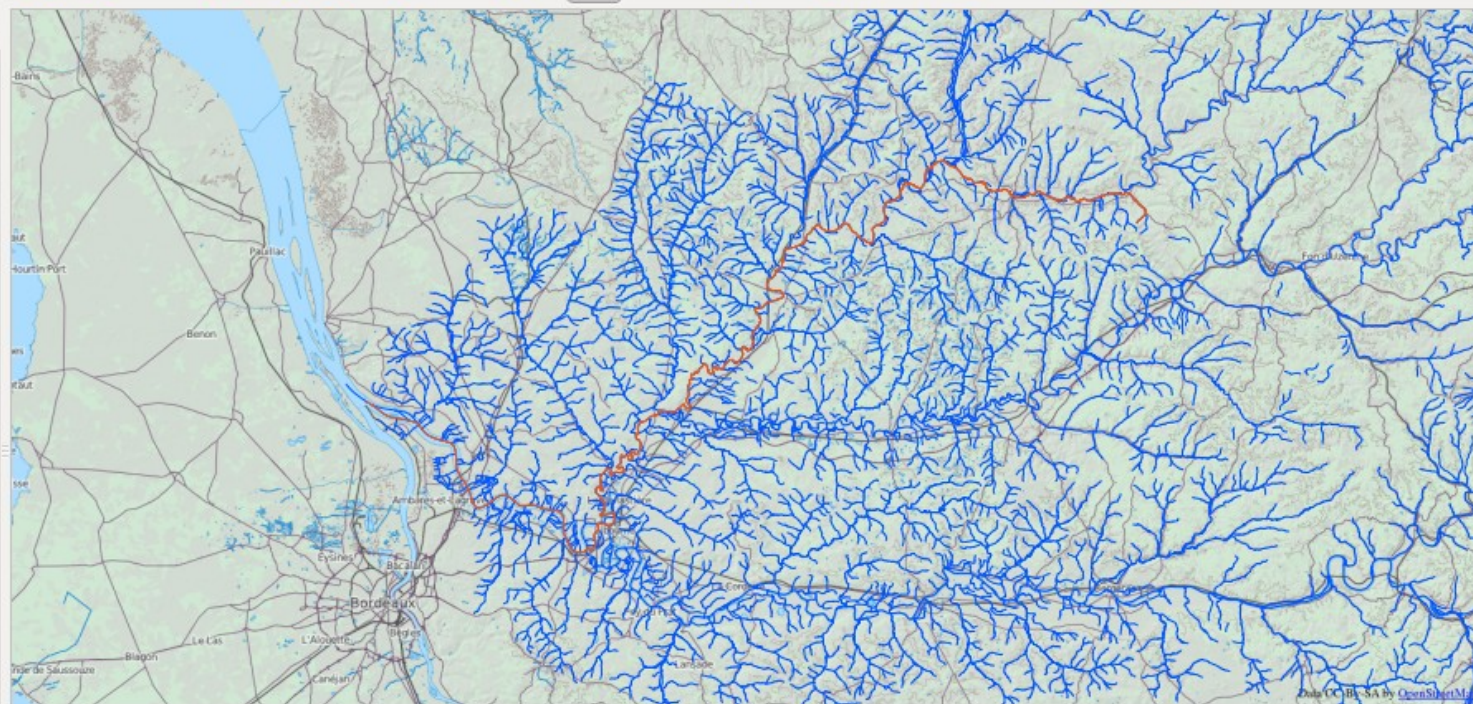
id integer	gid integer	the_geom geometry
1	173277	01020000206A0800002E0000
2	173278	01020000206A0800000F0000
3	173281	01020000206A080000070000
4	173280	01020000206A0800000D0000

Fichier Éditer Vue Couche Préférences Extension Vecteur Raster Base de donnée Aide



Couches

- ☒ shortest_path_topology
- ☐ shortest_path_pgrouting
- ☒ hydro network
- ☒ background



Query

`select * from dijkstra_sp('hydroedge_sp', 182521, 189555)`

local - topo

id

the_geom

Run

Get Layer

add layer

☒ Contrôle de l'ordre de rendu des couches

-124917.52,5579031.90 : 113856.43,5692234.41



Coordonnée :

-69795,5610492

Échelle

:1016276



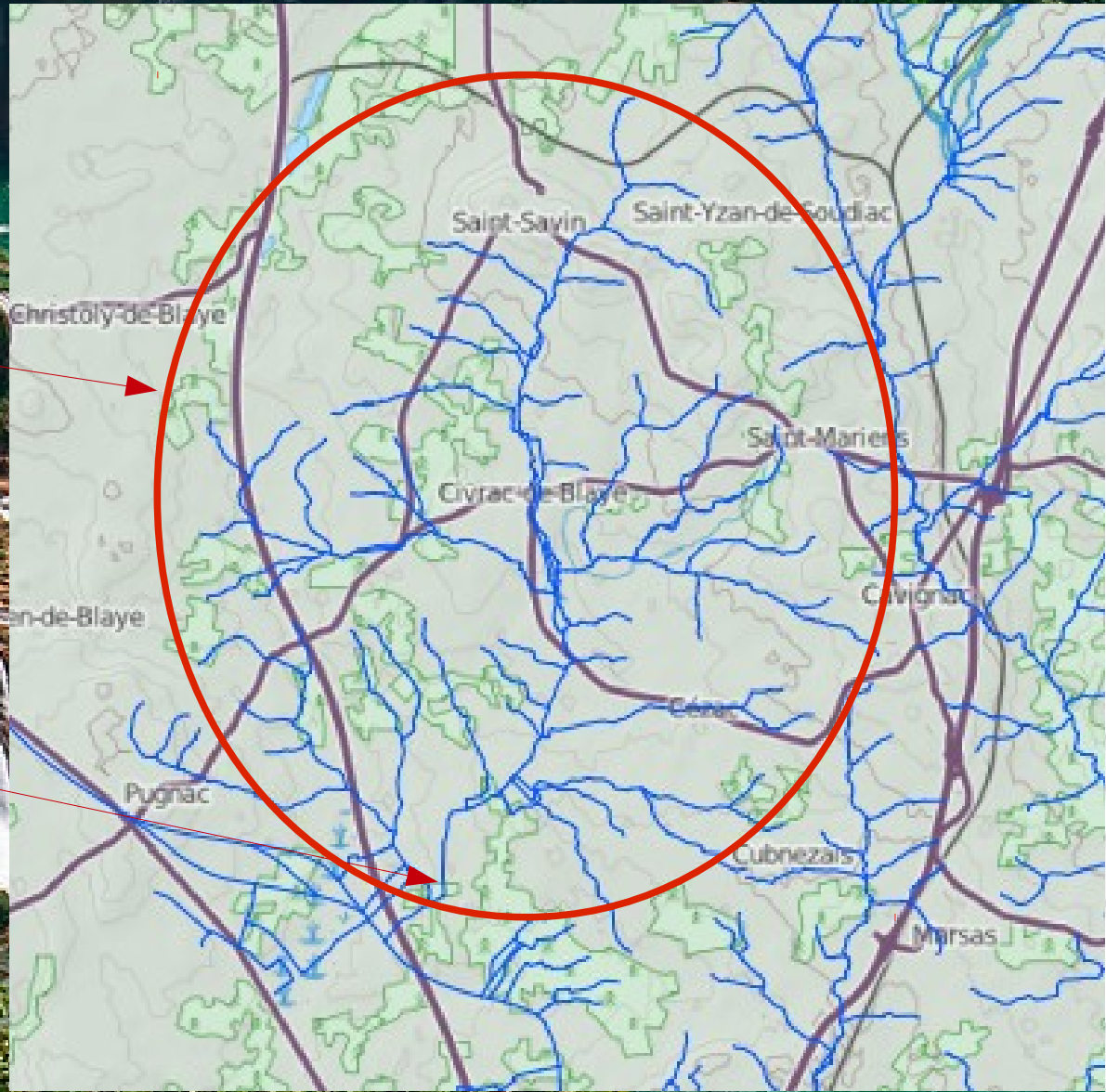
☒ Rendu

EPSG:900913

Find upstream

We want all connected edges upstream

Starting edge



Touching upstream edges

```
/* == Find all upstream edges == */
```

```
-- our starting edge : 31913
```

```
select gid, source, target, hname, cost from tr where gid = 31913;
```

```
/*
```

gid	source	target	hname	cost
31913	20850	21413	ruisseau le moron	2666.05230179502

```
*/
```

```
-- our starting edge and all upstream touching edges
```

```
select gid, source, target, hname, cost from tr where gid = 31913
```

```
union all
```

```
select gid, source, target, hname, cost from tr where target = 20850
```

```
/*
```

gid	source	target	hname	cost
31913	20850	21413	ruisseau le moron	2666.05230179502
33855	20735	20850	ruisseau de la marzelle	807.256330186324
32477	20845	20850	ruisseau le moron	59.7117241419599

```
(3 rows)
```

```
*/
```



create table

rec_res as

with recursive

search_graph(gid, source, depth, path, length, cycle) as (

```
select
    g.gid, g.source, 1 as depth, ARRAY[g.gid] as path
    , cost, false as cycle
from
    tr as g
where
    gid = 31913
```

1

union all

```
select
    g.gid
    , g.source
    , sg.depth + 1 as depth
    , path || g.gid as path
    , sg.length + g.cost as length
    , g.gid = ANY(path) as cycle
from
    tr as g
join
    search_graph as sg
on
    sg.source = g.target
where
    not cycle
```

2

Recursive CTE

select

```
sg.*
, tr.geom
```

from

```
search_graph as sg
```

join

```
tr
```

on

```
sg.gid = tr.gid
```

limit 1000;

3

gid	source	depth	path	length	cycle	geom
integer	integer	integer	integer[]	double precision	boolean	geometry(MultiLineString,2154)
31913	20850	1	{31913}	2666.0523017	f	01050000206A080000001000
33855	20735	2	{31913, 3473}	3086319	f	01050000206A080000001000
32477	20845	2	{31913, 2725}	7640259	f	01050000206A080000001000
33854	19909	3	{31913, 7183}	7295195	f	01050000206A080000001000

1 : init

```
select
    g.gid, g.source, 1 as depth, ARRAY[g.gid] as path
    , cost, false as cycle
from
    tr as g
where
    gid = 31913
```

2 : recursive part

select

```
g.gid  
  , g.source  
  , sg.depth + 1 as depth  
  , path || g.gid as path  
  , sg.length + g.cost as length  
  , g.gid = ANY(path) as cycle
```

Stack the gid to the path
for this record

Sum up the cost
(it's the length here)

If the record gid is already
in the path, we have a cycle

from

```
tr as g
```

join

```
search_graph as sg
```

on

```
sg.source = g.target
```

where

```
not cycle
```

Join result set from
previous iteration
to connected upstream
edges

Do not take elements
which make a cycle

3 : Get results






```
select
    sg.*
    , tr.geom
from
    search_graph as sg
join
    tr
on
    sg.gid = tr.gid
limit 1000;
```

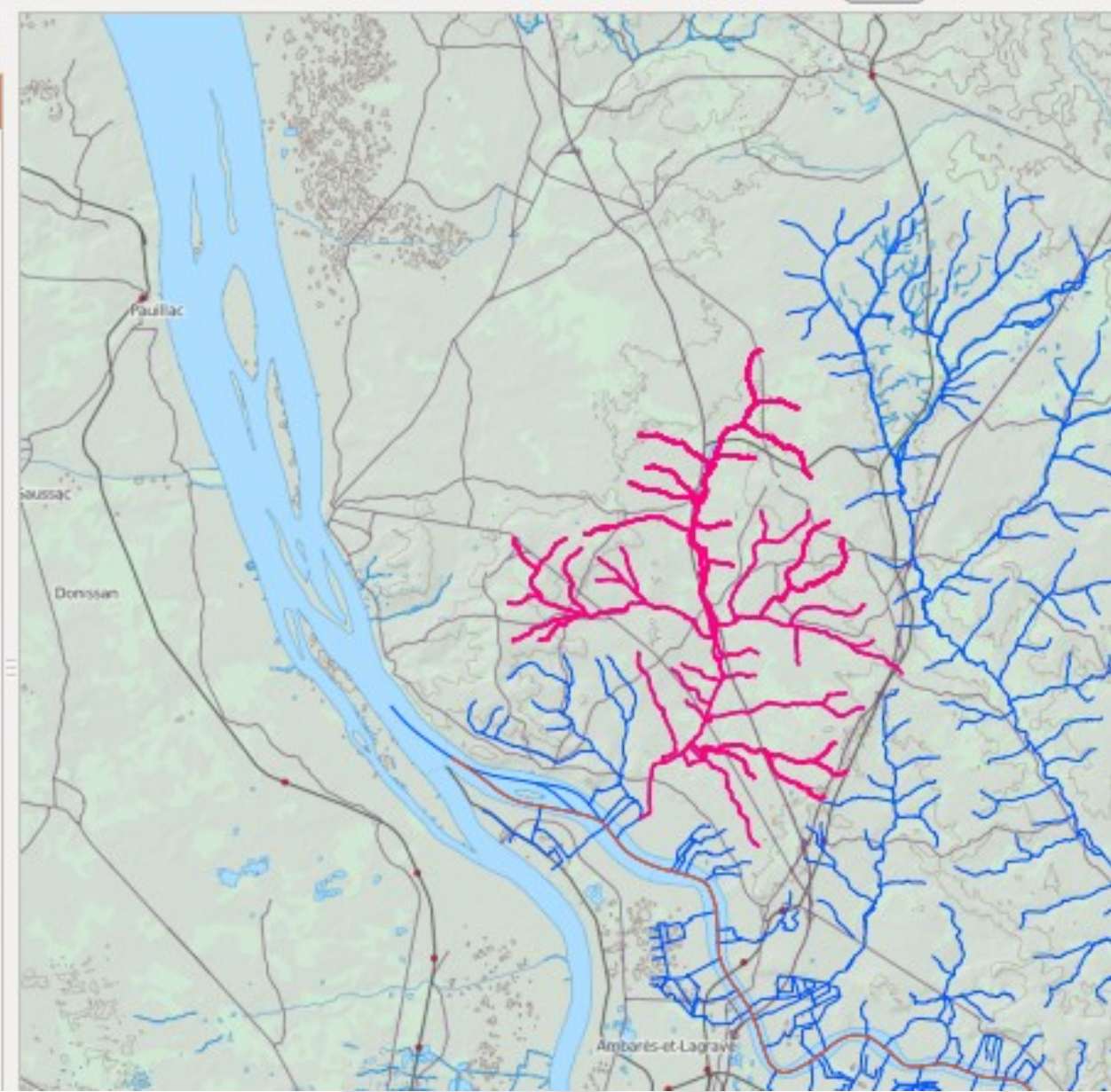
Join CTE results to original table to get geometries

Better limit recursive queries to avoid unfinite loops

gid integer	source integer	depth integer	path integer[]	length double precision	cycle boolean	geom geometry(MultiLineString,2154)
31913	20850	1	{31913}	2666.0523017	f	01050000206A080000001000
33855	20735	2	{31913,3473}	3.3086319	f	01050000206A080000001000
32477	20845	2	{31913,2725}	7.7640259	f	01050000206A080000001000
33854	19909	3	{31913,7183}	7.7295195	f	01050000206A080000001000

Couches

- ▶ ☒  recursive_upstream
- ▶ ☒  shortest_path_topology
- ▶ ☐  shortest_path_pgouting
- ▶ ☒  hydro network
- ☒  background



... and on PostGIS topology

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

)

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;

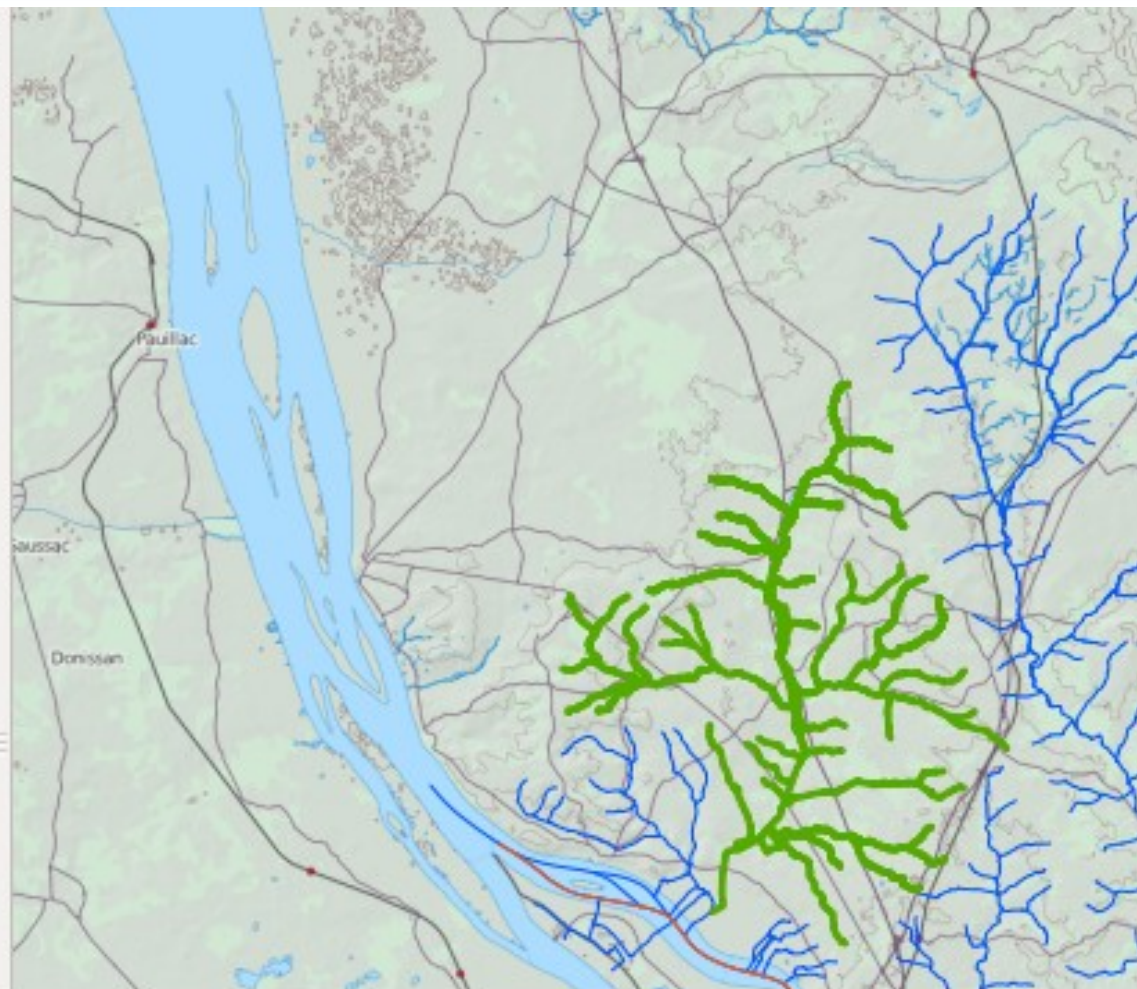
1

2

3

Couches

- ▶ ☒  recursive_upstream_topo
- ▶ ☒  recursive_upstream
- ▶ ☒  shortest_path_topology
- ▶ ☐  shortest_path_pgrouting
- ▶ ☒  hydro network
- ▶ ☒  background



Attribute table - recursive_upstream_topo :: 0 / 478 feature(s) selected

	edge_id ▲	start_node	depth	path	length	cycle
0	173832	189333	1	{173832}	2666.05230...	f
1	173452	189332	2	{173832,17...	3473.30863...	f



Time out, brains washed, that's the end

Thanks !

Any question ?

vincent.picavet@oslandia.com

Twitter : @vpicavet

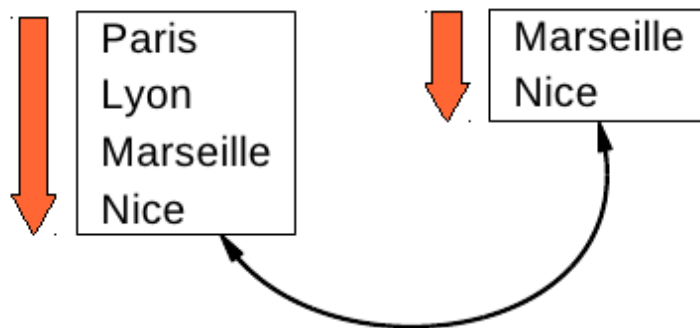
<http://www.oslandia.com>

BONUS SLIDES

GIS : use case



- Bisonvert.net
 - Car-sharing free software
- Goal
 - match people doing the same journey
- Current method
 - Match from/to via names



GIS : use case

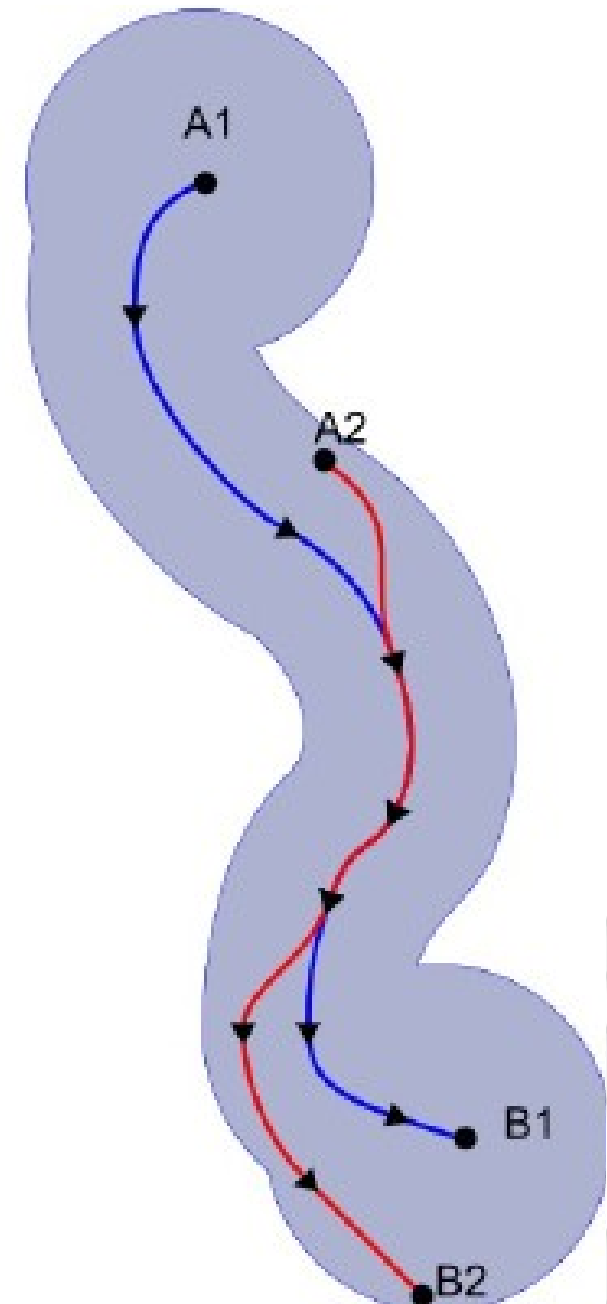
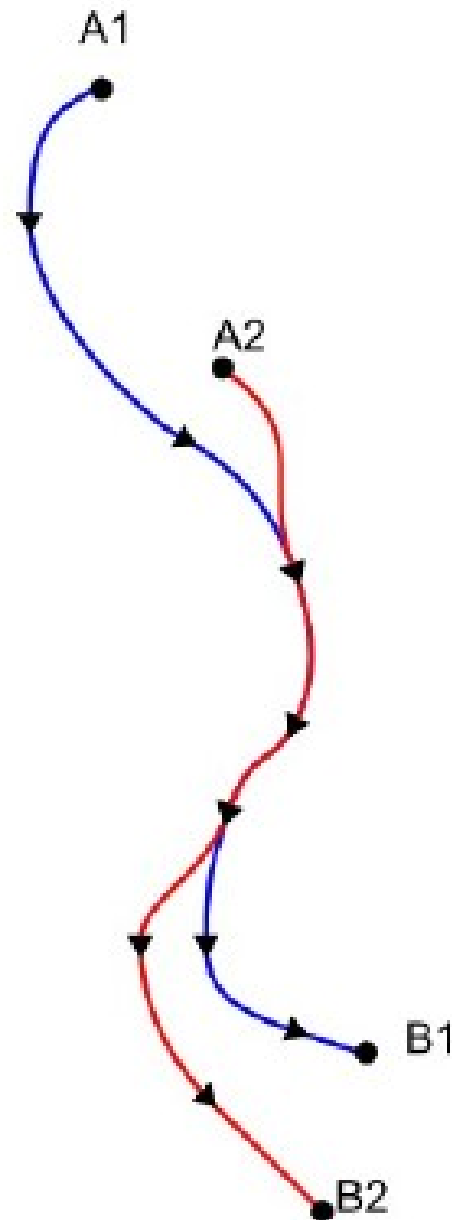
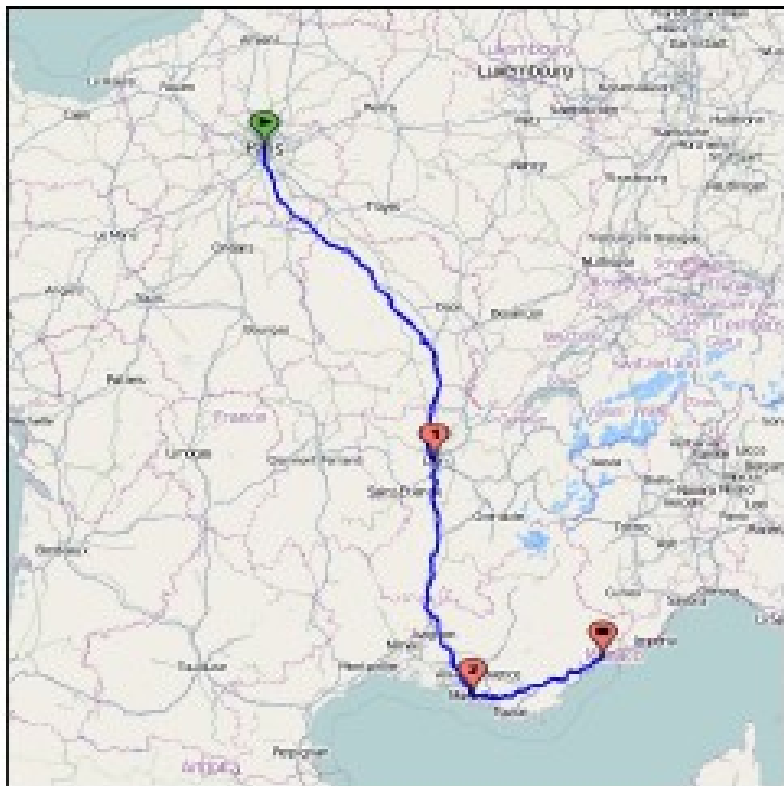
Solution :

Use real paths

1/ Compute path (routing)

2/ Match paths

(Spatial analysis)




```
/* === Compute a sum of numbers from 1 to 1000 === */  
  
-- CTE only has one field  
WITH RECURSIVE t(n) AS (  
    -- Initialization : just one value  
    VALUES (1)  
    -- recursive part : for every element of previous recursion  
    -- we take value n+1  
    -- only if n is below 100  
    UNION  
        SELECT n+1 FROM t WHERE n < 100  
)  
-- get all results|  
SELECT n FROM t;
```