PostGIS 2.x

Olivier Courtin

PostgreSQL Session #6 - 2014 – Paris

PostGIS spatial database

2.0.0	04/2012
2.1.0	08/2013

Current version: 2.1.4

Coming 2.2

Management

Advanced spatial analysis

Topology

Raster

Point Cloud

3D

Management

Advanced spatial analysis

Topology

Raster

Point Cloud

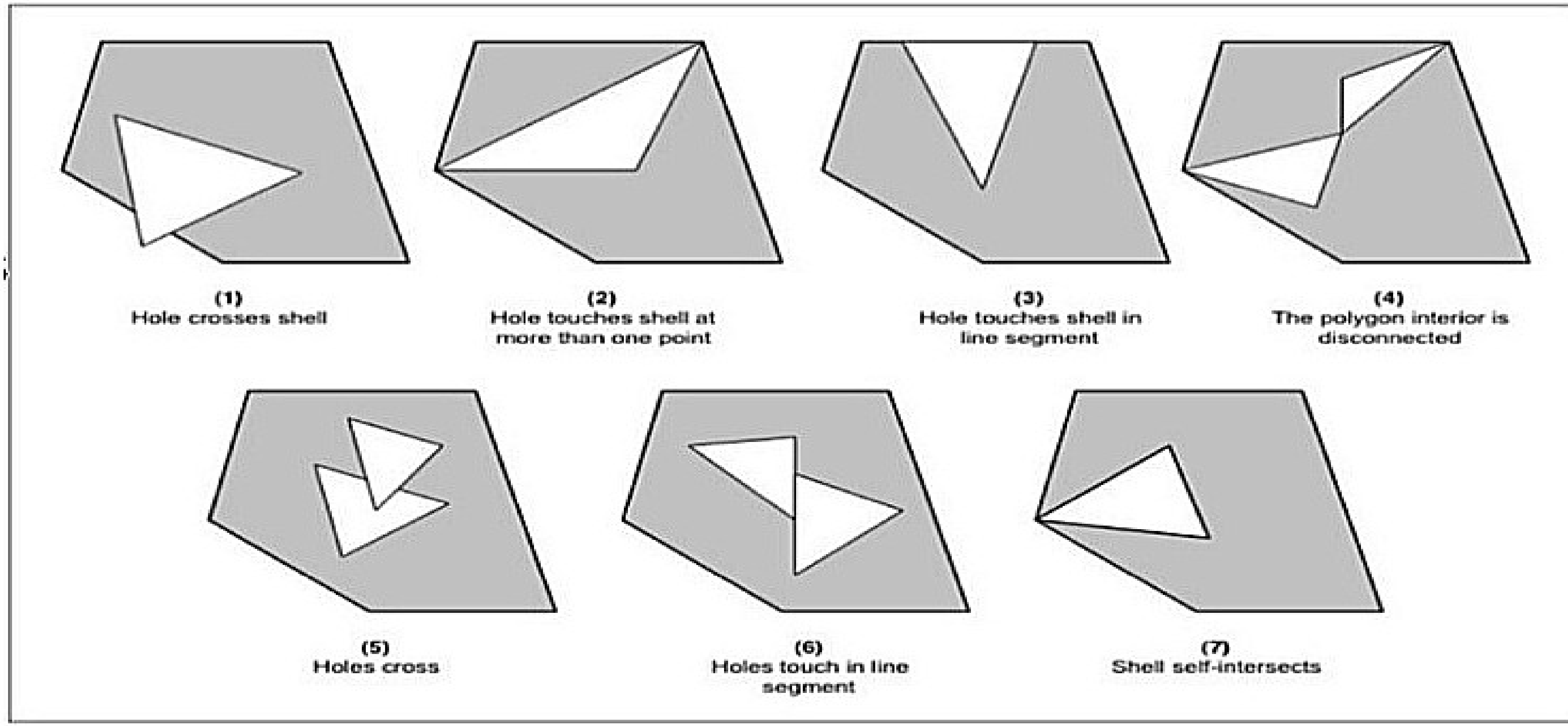
3D

```
CREATE EXTENSION postgis ;
CREATE EXTENSION postgis_topology ;
```

geometry_columns (and geography_columns) are now views (rather than table)

```sql
CREATE TABLE buildings (
        gid SERIAL PRIMARY KEY
        , geom geometry(MultiPolygon, 26986)
);

alter table buildings
        alter column geom
                type geometry(MultiPolygon, 2154)
                using st_setsrid(geom, 2154);
```



examples of invalid multipolygon

UPDATE my_schema.my_table
SET geom =
ST_CollectionExtract(**ST_MakeValid**(geom), 3);

Management

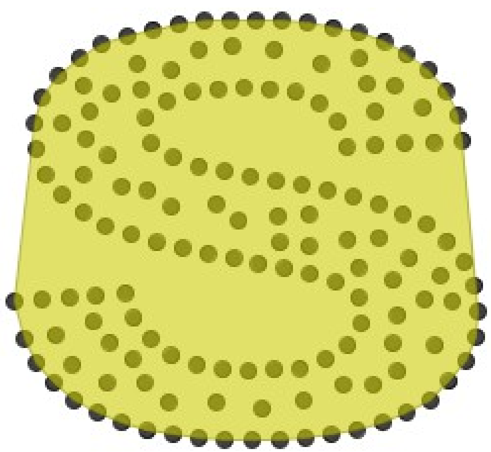
Advanced spatial analysis

Topology

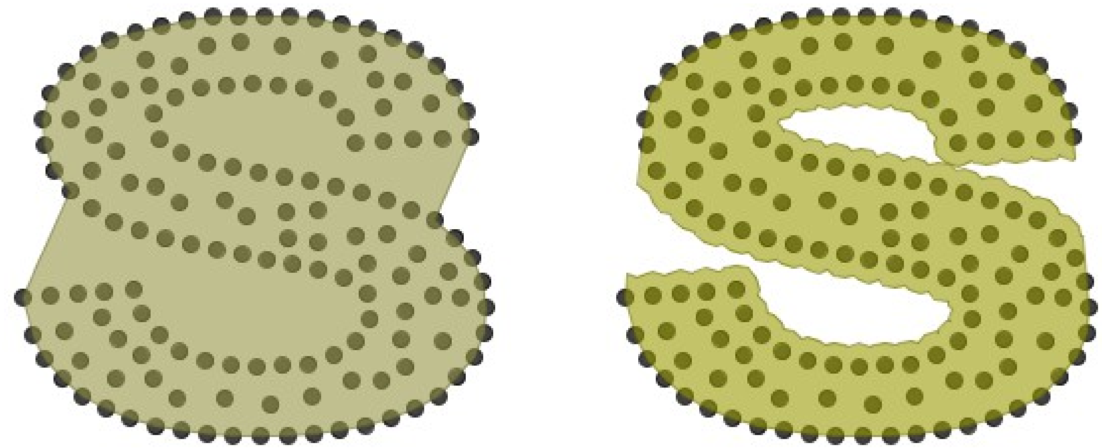
Raster

Point Cloud

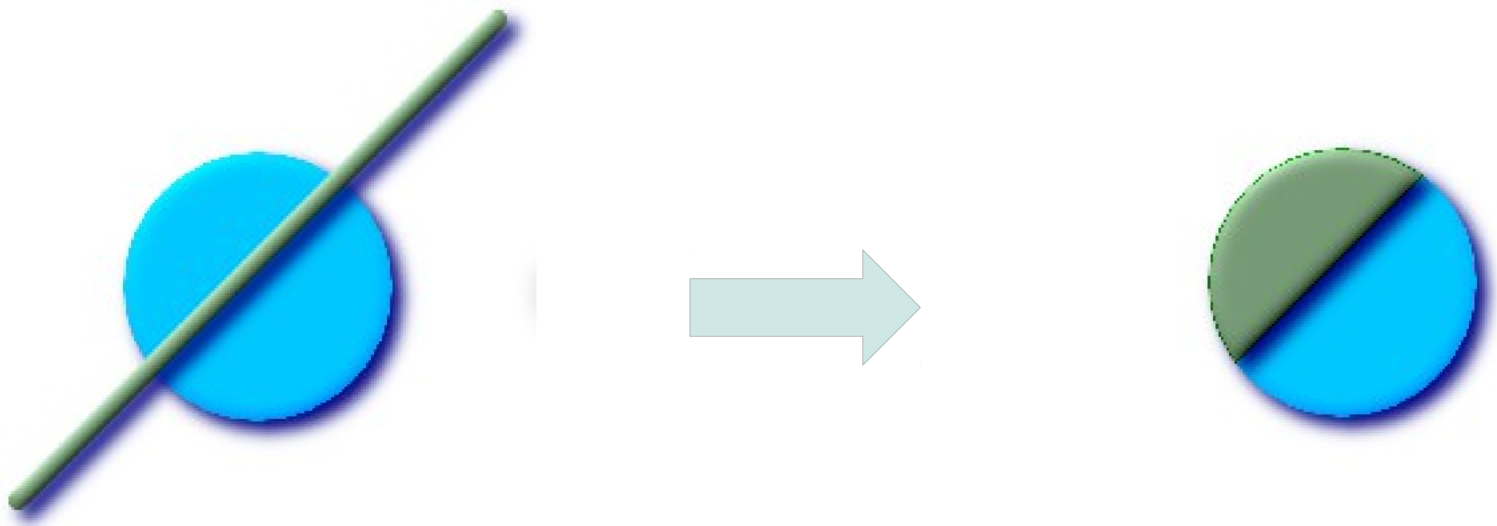
3D



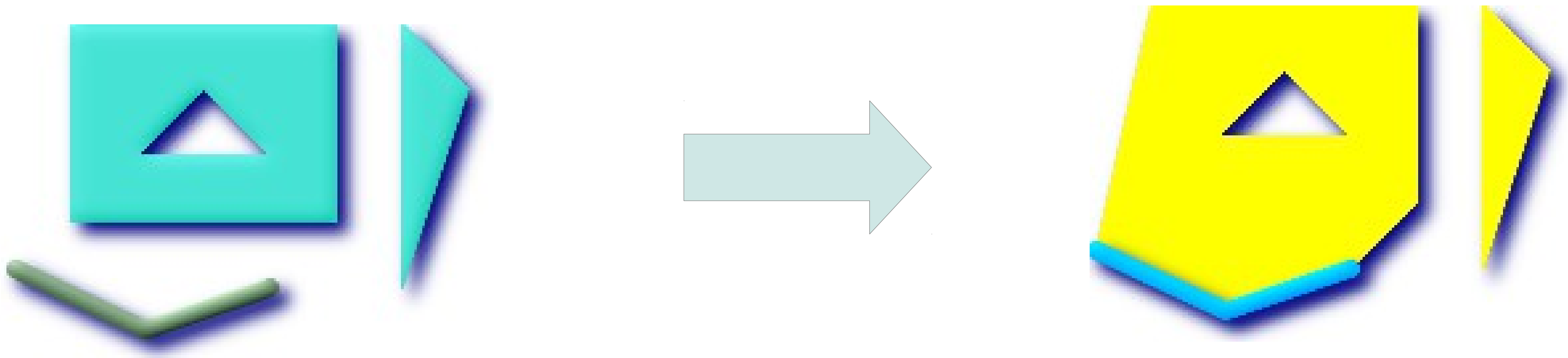
ST_ConvexHull



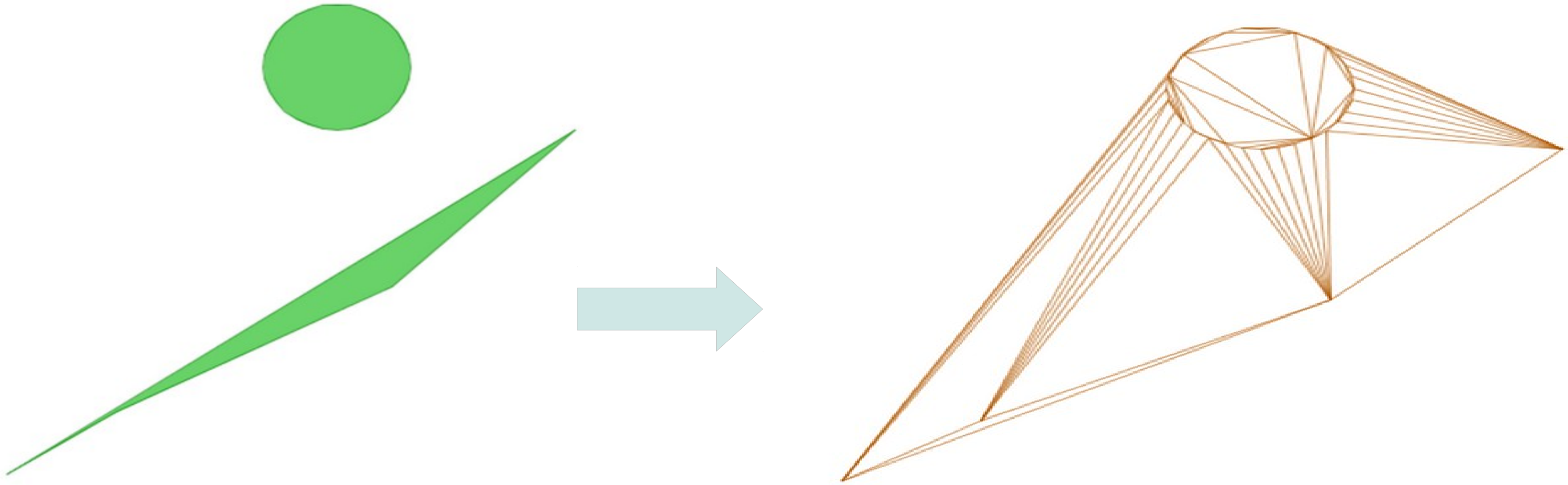
ST_ConcaveHull



ST_Split



ST_Snap



```
SELECT
        ST_DelaunayTriangles(
                ST_Union(ST_GeomFromText('POLYGON((175 150, 20 40,
                        50 60, 125 100, 175 150))'),
                ST_Buffer(ST_GeomFromText('POINT(110 170)'), 20)
                ),0.001,1)
         As  dtriag;
```

KNN-GIST: Spatial nearest neighbors

```
SELECT name, gid FROM geonames
ORDER BY geom <->
ST_SetSRID(ST_MakePoint(-90,40),4326)
LIMIT 10;
```

Distance operator: <-> or <#> (center or bbox)

Management

Advanced spatial analysis

Topology

Raster

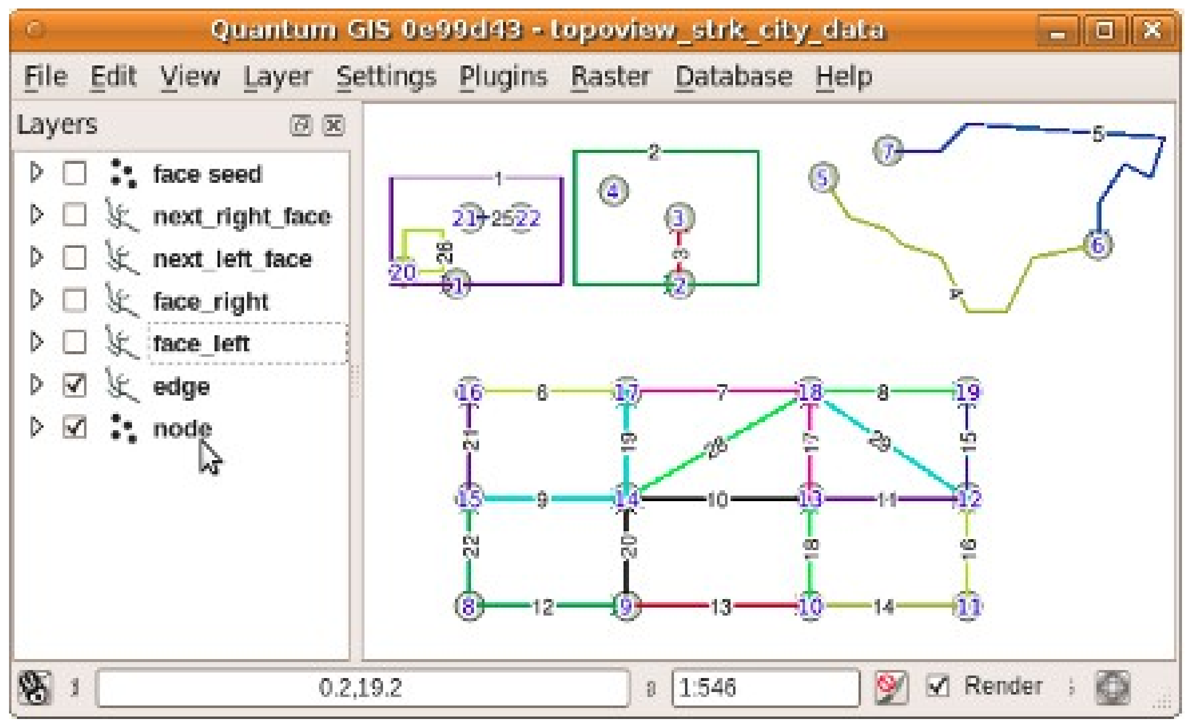
Point Cloud

3D

Topology

node / edge / face model

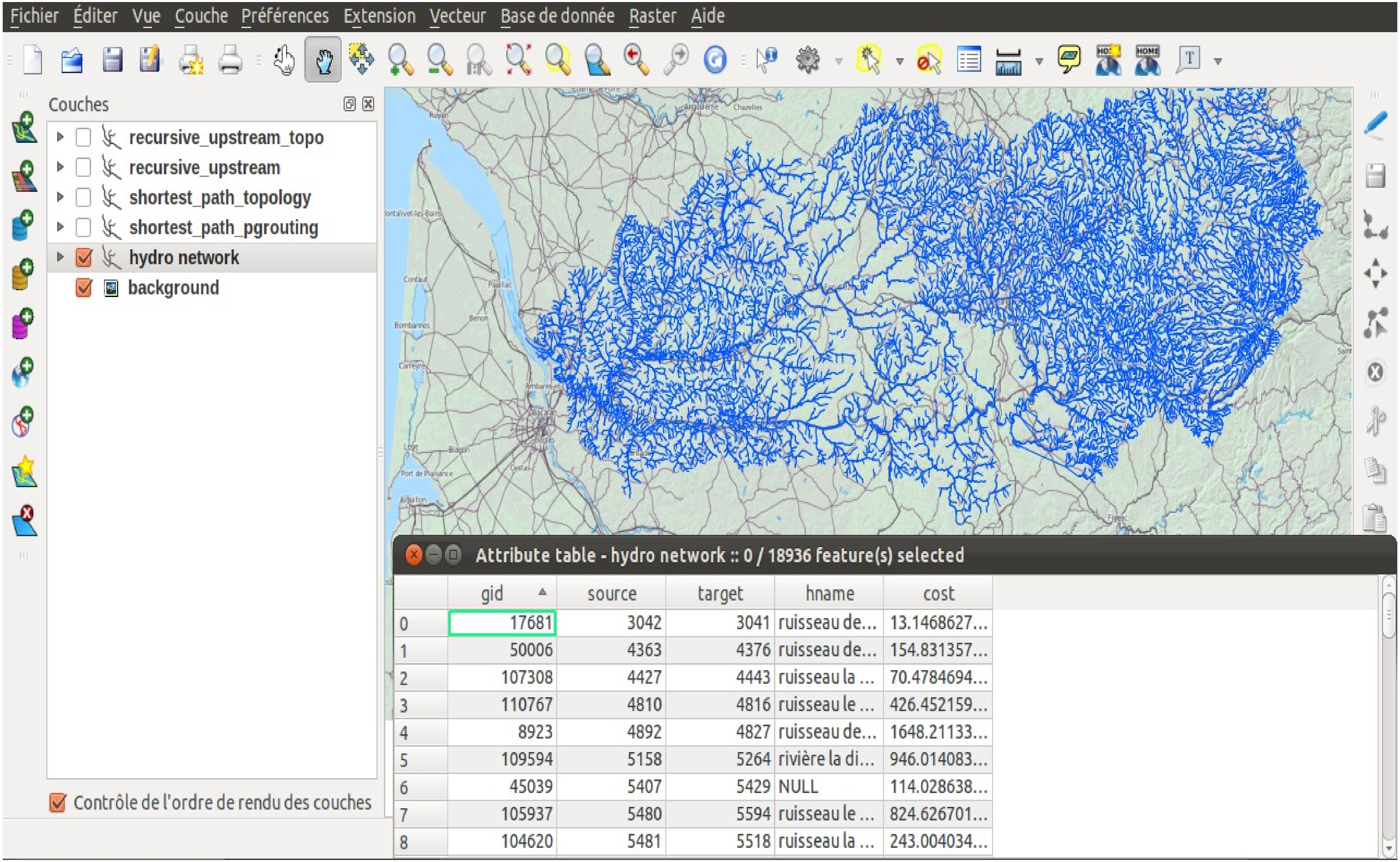
ISO SQL/MM functions



Source: Sandro Santilli

Using ST_Simplify **without** topology



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
Contrôle de l'ordre de rendu des couches
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...

```sql
-- 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;
```

```sql
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,2154)
1	175256	190369	190361	175230	-175243	0	0	01020000206A0800000
2	167356	183762	181917	166725	167356	0	0	01020000206A0800001

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

eau sortie

rtie de données | Expliquer (Explain) | Messages

	gid integer	topogeom topology.topogeometry
	116768	(1,1,163704,2)
	116767	(1,1,163705,2)
	116765	(1,1,163706,2)

Recursive CTE

```
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
                )
```

1

2

3

```
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;
```

```sql
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
)
```

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

Join result set from previous iteration to connected upstream edges

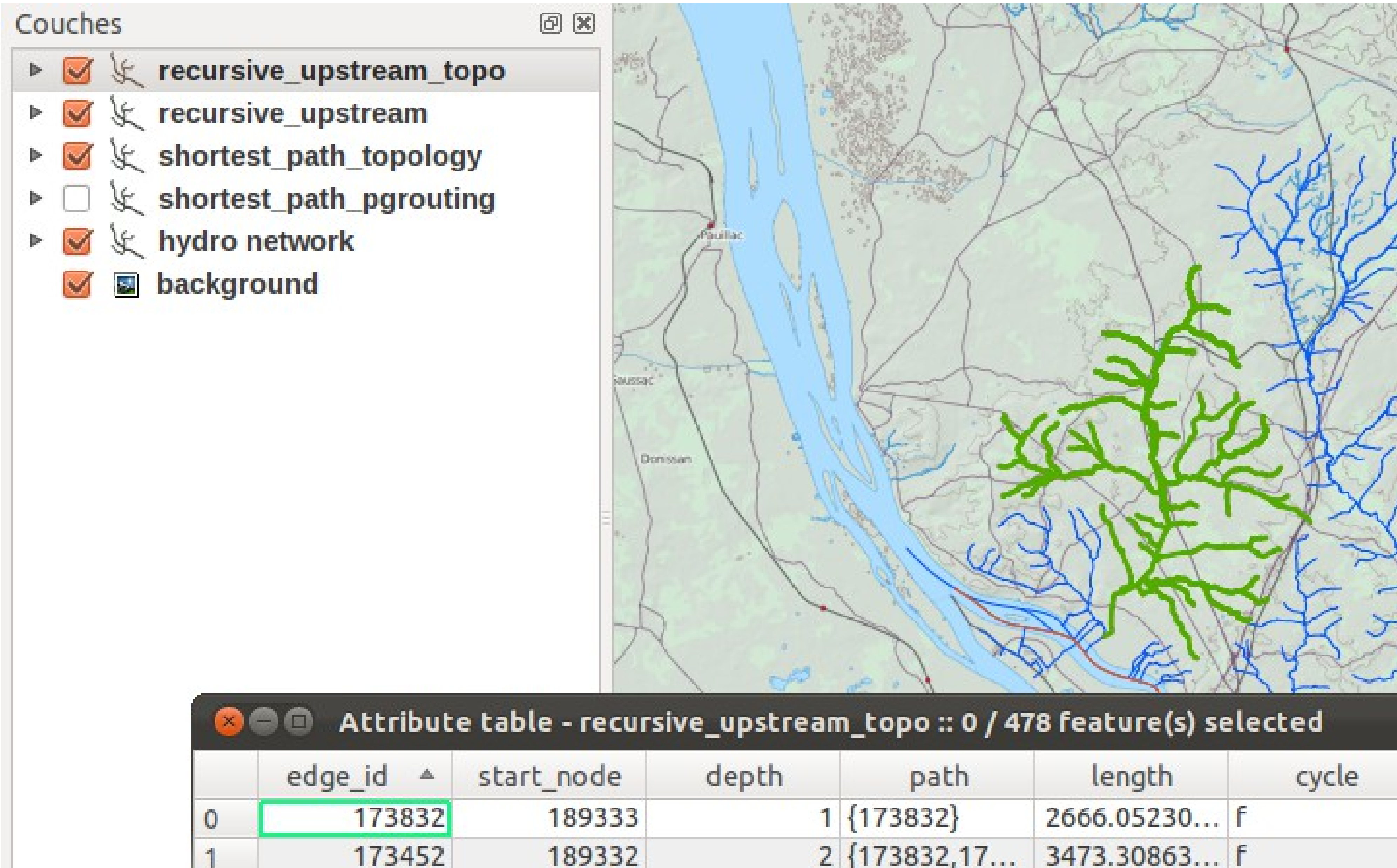
Do not take elements which make a cycle

```sql
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;
```

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	01050000206A08000001000
33855	20735	2	{31913,	3473.3086319	f	01050000206A08000001000
32477	20845	2	{31913,	2725.7640259	f	01050000206A08000001000
33854	19909	3	{31913,	7183.7295195	f	01050000206A08000001000


Couches
recursive_upstream_topo
recursive_upstream
shortest_path_topology
shortest_path_pgrouting
hydro network
background
Pauillac
Donissan
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

Management

Advanced spatial analysis

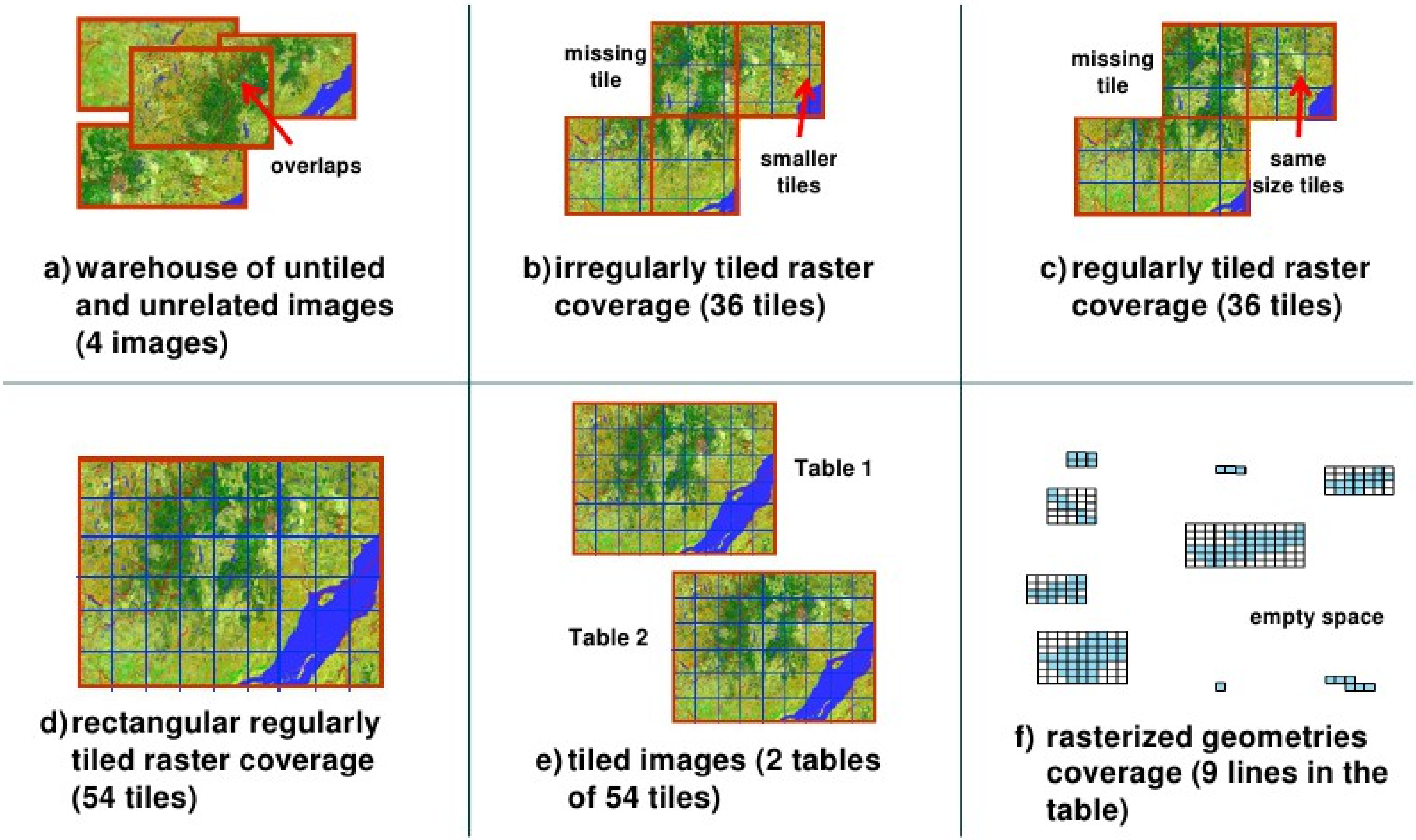
Topology

Raster

Point Cloud

3D

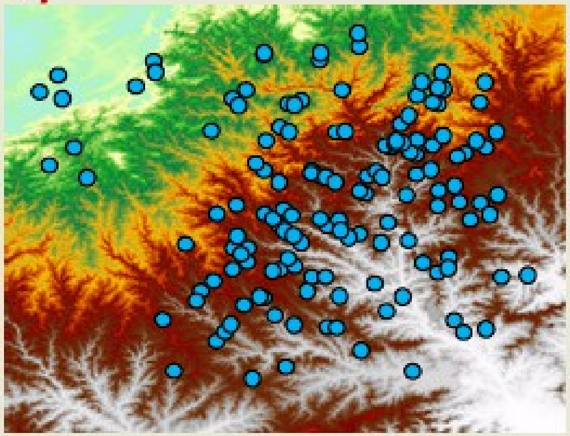
- Raster / vector analysis
- New raster datatype (using tiles)
- Multiresolution, multiband, tile coverage
- Import/export (GDAL)
- Raster functions



Source: Pierre Racine

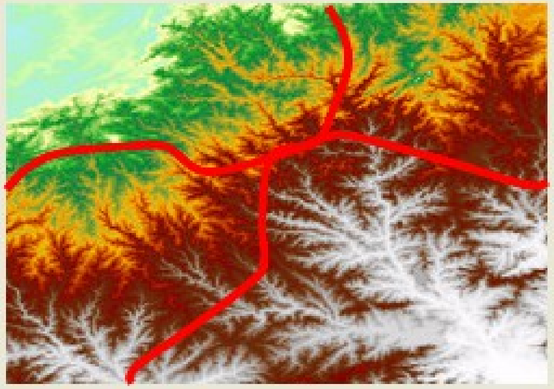
Extract ground elevation values for lidar points…

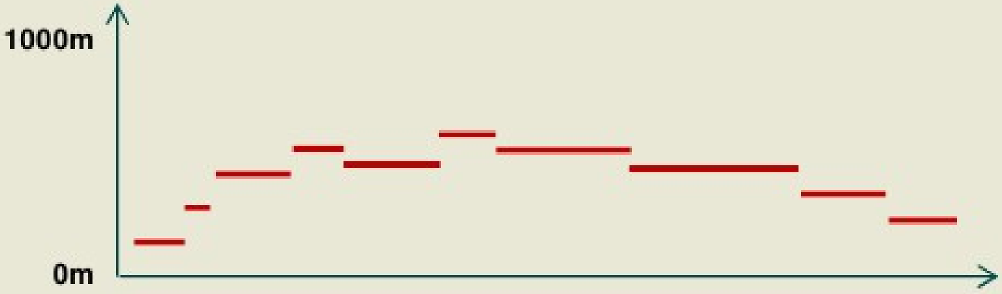
- SELECT pointID, ST_Value(rast, geom) elevation
  FROM lidar, srtm WHERE ST_Intersects(geom, rast)



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)





Source: Pierre Racine

Management

Advanced spatial analysis

Topology

Raster

Point Cloud

3D

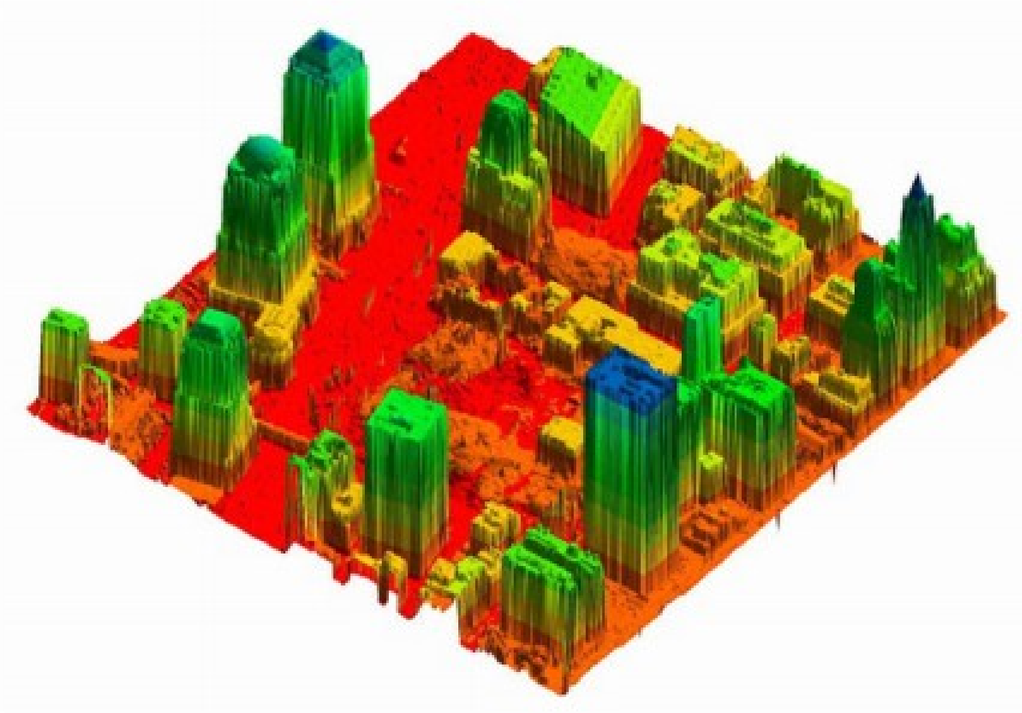
As PostgreSQL and PostGIS extension

Handle Patches

Arbitrary dimension handling

Data compression

PDAL (as a loader)



Management

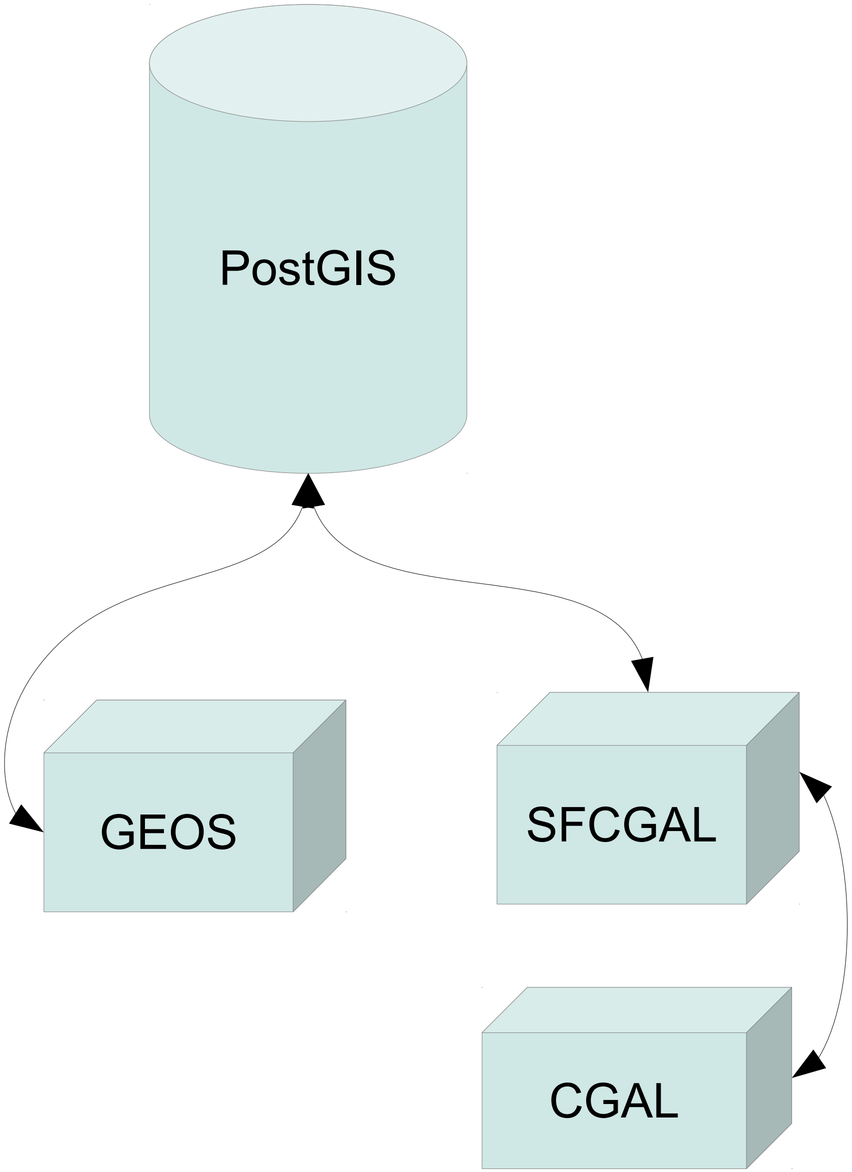
Advanced spatial analysis

Topology

Raster

Point Cloud

3D


PostGIS
GEOS
SFCGAL
CGAL

CGAL

ST_3DIntersection

ST_Tesselate

ST_3DArea

ST_Extrude

ST_ForceLHR

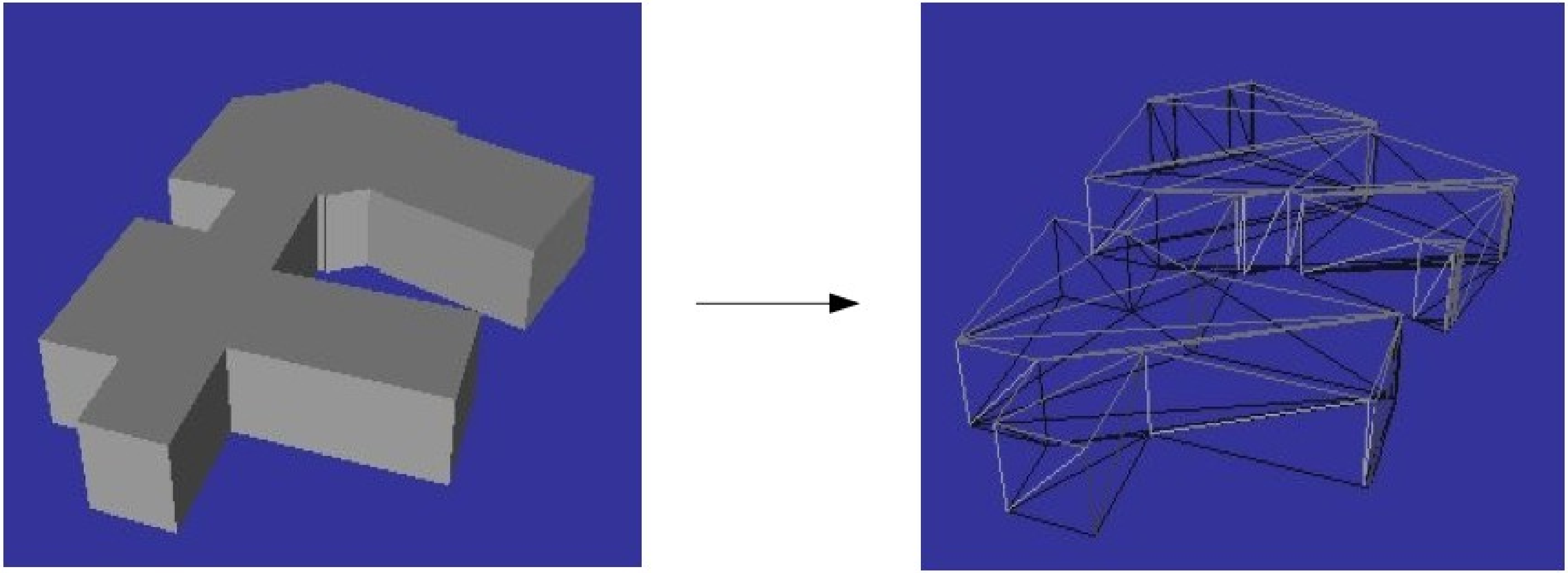
ST_Orientation

ST_MinkowskiSum

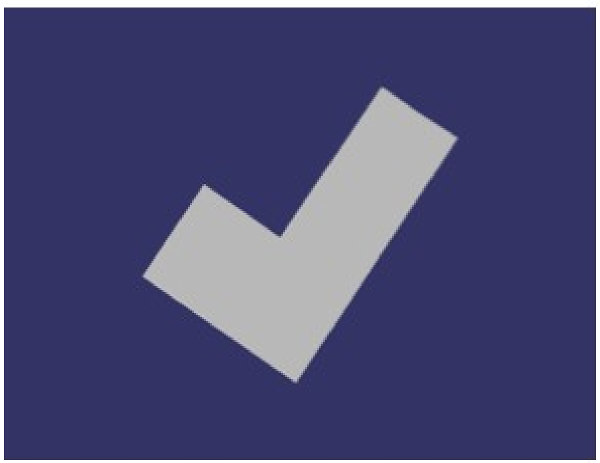
ST_StraightSkeleton

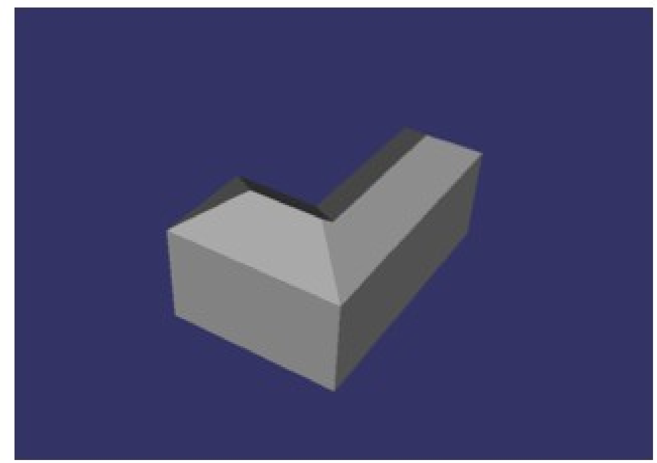
SFCGAL functions

ST_Tesselate



ST_StraightSkeleton





2D Building
Footprint

Straight Skeleton

Extrusion
& roof computation

ST_Intersects

ST_3DIntersects

ST_Intersection

ST_Area

ST_Distance

ST_3DDistance

Both GEOS & SFCGAL

```
SET postgis.backend = 'geos' ;

SET postgis.backend = 'sfcgal' ;
```

SFCGAL perfomances similar to GEOS ones for 2D

(but with SFCGAL we gain arbitrary precision)

SFCGAL perfomances similar to GEOS ones for 2D

(but with SFCGAL we gain arbitrary precision)

But some 3D computation could take time.

https://vimeo.com/74869530

https://vimeo.com/105323534

www.oslandia.com

http://www.postgresql-sessions.org/6/start