Advanced spatial analysis with PostGIS

San Francisco

Hyatt San Francisco Airport
March 9 - 12, 2015

UNIVERSITÉ LAVAL

cef
Centre d'étude de la forêt

Pierre Racine
Research assistant
GeoElucubrations

Advanced Query Examples

1) Vector/Vector analyses
   - Extraction from a polygon coverage for points
   - Extraction from a polygon coverage for polygons
2) Vector/Raster analyses
   - Extraction from a raster coverage for points
   - Extraction from a raster coverage for polygons
3) Elevation profiles
4) Proximity analyses
   - N nearest points from one point
   - N nearest points for each point of a table
   - N nearest geometries for each geometry of a table
5) Raster/Raster analyses
   - Map algebra
   - Union of overlapping rasters
6) Rasterization of a vector coverage
7) Overlap removal in a vector coverage

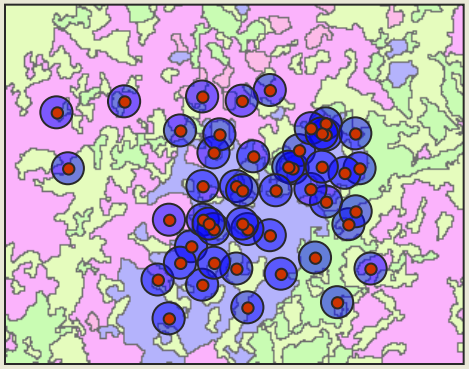
PostGIS Addons

- A single SQL file of pl/pgsql functions
  - With tests and uninstall script
  - Goal: Make easy sharing of users contributed PostGIS functions
- Notably:
  - **ST_ExtractToRaster()** to extract any metric from a vector coverage into a raster. e.g. number of points, value of most biggest polygons, and many more… Good for rasterizing a vector layer.
  - **ST_RandomPoints()** to generate random points within a polygon.
  - **ST_AreaWeightedSummaryStats()** and **ST_SummaryStatsAgg()** to aggregate stats resulting from vector/vector and raster/vector intersections.
  - **ST_DifferenceAgg()** to remove overlaps in a polygon coverage.
  - **ST_CreateIndexRaster()** to create a raster having a unique value per pixel.
  - **ST_AddUniqueID()** to quickly add a unique identifier column.
  - **ST_BufferedSmooth()** to smooth a geometry by dilatation/erosion.

Have a look at Geospatial Elucubrations!

1) Extraction from a vector coverage for polygons

- Extract, for a series of points, lines or polygons, underlying values from another vector coverage.
- e.g. which type of land cover a series of polygons intersect with



observ	
geom	obsid
polygon	24
polygon	31
polygon	45
…	…

∩

cover	
geom	ctype
polygon	4
polygon	3
polygon	5
polygon	2
…	…

=

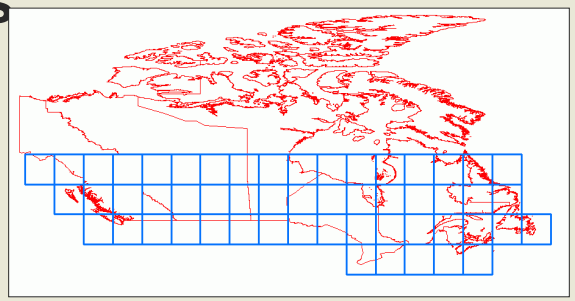
result			
geom	obsid	ctype	area
polygon	24	4	10.34
polygon	53	3	11.23
polygon	24	5	14.23
polygon	23	2	9.45
…	…	…	…

```
SELECT obsid, ctype, ST_Area(geom) area, geom
FROM (SELECT ST_Intersection(o.geom, c.geom) geom, obsid, ctype
        FROM observations o, couvert c
        WHERE ST_Intersects(o.geom, c.geom)) foo;
```

The RASTER type

- Main addition to PostGIs 2.0.x
- A raster is generally splitted (tiled) over many table rows
  - limit of 1GB per row, theoretically 32GB coverage
- Each tiles is georeferenced and spatially indexed
  - width, height, upperleftx, upperlefty, scalex, scaley, skewx, skewy, srid
  - tiles can overlaps or be sparce
- Each raster (or tile) can have many bands
  - pixel type, nodata value
- Handle overviews in sister tables
  - lower resolutions for fast display
- Compressed by PostgreSQL (very good!)
- The raster_column view hold the list of tables having a raster column along with their metadata

e.g. SRTM Coverage for Canada



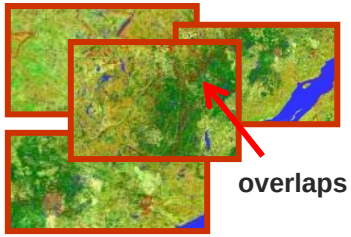
Why store raster in the database?

- One simple language for everything: SQL
  - Many raster functions are similar to vector ones…
  - Complex spatial analyses can be done with **a single** SQL query.
- Raster/Vector interaction
  - Your vector data are normally **already** in the database…
- Performance
  - Analysis processing is generally **faster** on tiled raster coverage.
- Data volume
  - You can work on **TB** raster coverages without much problems.
- You can even keep the raster's data outside the database…
  - …and use them transparently inside the db with SQL
  - Only metadata are stored inside (extent, SRID, pixel type, nodata) Pixel values area read from the referenced files via GDAL.
  - raster2pgsql –R option

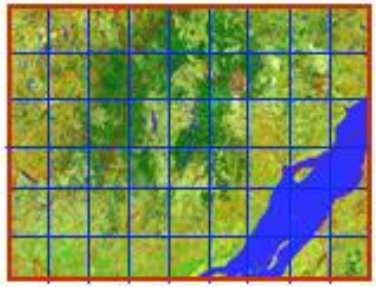
Rasters vs Tiles

- Even if we speak often about “tiles” when speaking about raster in PostGIS, there is no “tile” object, only rasters!
  - A tile is simply a way to view a raster when it is part of a coverage.
  - All tiles are regular rasters. Rasters are not necessarily tiles…
- Advantages
  - Simplicity - Only one concept to understand – Simple schemas.
  - You can store any irregular raster arrangements
    - Tiles having different sizes, overlapping tiles, missing tiles, etc…
    - Just load your messy raster coverage and you’re ready to work!
- You must take this into account in your queries…
  - Add ST_Intersects(rast, geom) in most raster/vector queries.
  - Aggregate tile stats when querying stats for a whole coverage.
  - Sometimes you must ST_Union(rast) some tiles together before processing them further (reprojecting, computing stats, etc…)

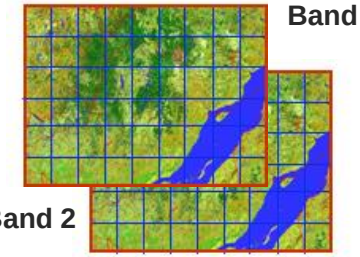
Possibles RASTER Arrangements



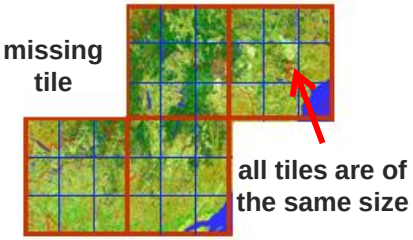
a) Satellite or aerial imagery warehouseou (4 images, 4 rows)



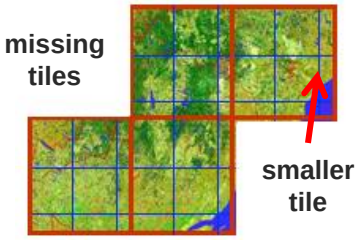
b) Regularly tiled rectangular raster coverage (54 tiles, 54 rows)



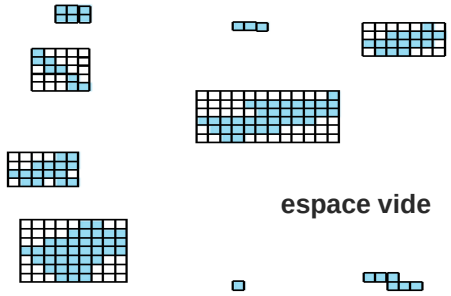
c) Multi-band tiled images (1 table of 108 tiles)



d) Regularly tiled raster coverage (36 tiles, 36 rows)



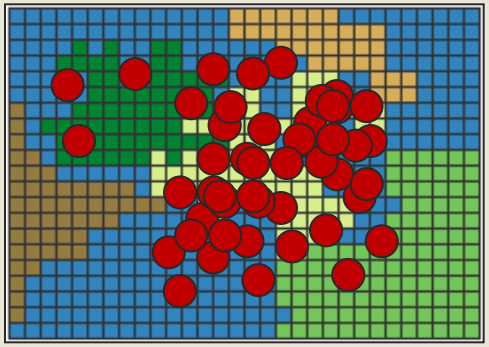
e) Irregularly tiled raster coverage (36 tiles, 36 rows)



f) A table of rasterized geometries (9 rasters, 9 rows)

2) Extraction from a raster coverage for points and polygons

- Extract, for a series of points, lines or polygons, underlying values from a raster coverage.
- e.g. compute mean temperature for a series of polygons



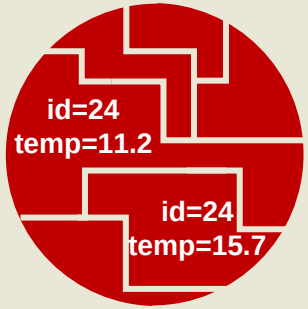
buffers	
geom	pointid
polygon	24
polygon	46
polygon	31
polygon	45
…	…

∩

temperature
raster
raster
raster
raster
raster
…

=

result		
geom	pointID	temp
polygon	24	11.2
polygon	53	13.4
polygon	24	15.7
polygon	23	14.2
…	…	…



Vector Mode (pixels are cut)

```
SELECT bufid,
(ST_AreaWeightedSummaryStats(gv)).*
FROM (SELECT ST_Intersection(rast, geom) gv
        FROM temperature, buffer
        WHERE ST_Intersects(rast, geom)) foo
GROUP BY bufid;
```

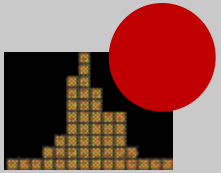
- when polygons size much smaller than pixel size
- or vector coverage is composed of lines
- slower, more precise

Raster Mode raster (pixels are not cut)

```
SELECT bufid,
(ST_SummaryStatsAgg(ST_Clip(rast, geom, true))).*
FROM temperature, buffer
WHERE ST_Intersects(rast, geom)
GROUP BY bufid;
```

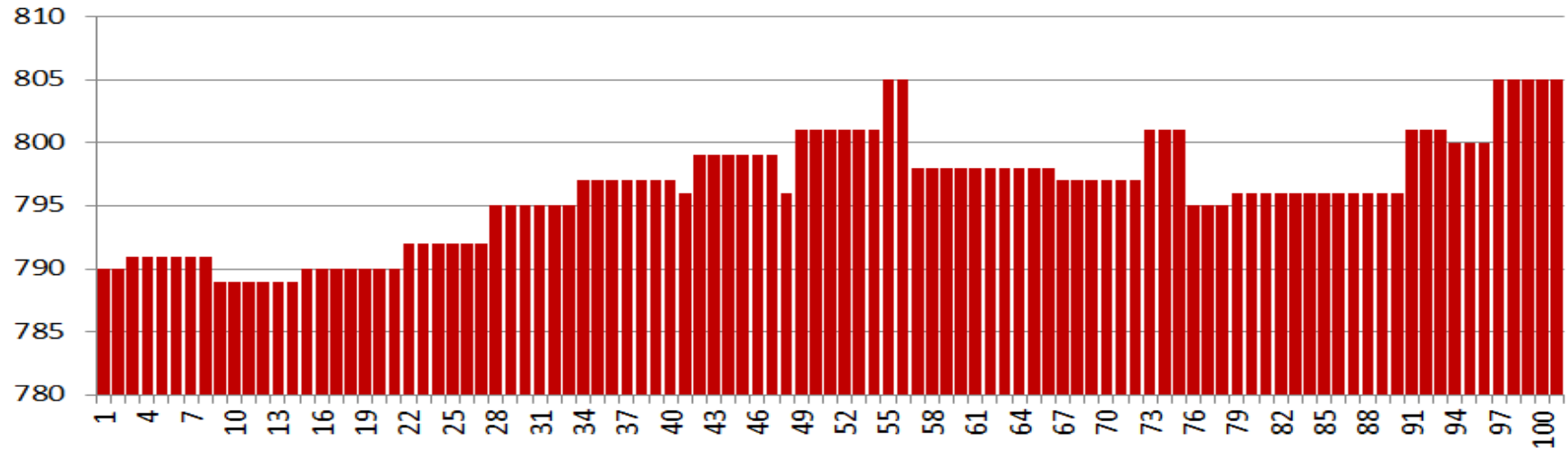
- when polygon size much bigger than pixel size
- good only for polygons coverage
- faster, less precise

Intersects ignore nodata values



3) Elevation profile

```
WITH points AS (
   SELECT id, ST_LineInterpolatePoint(geom, id/100.0) geom
   FROM generate_series(0, 100) id,
            (SELECT (ST_Dump(geom)).geom
             FROM linetable WHERE id = 2058) foo
)
SELECT id, geom, ST_Value(rast, geom) elev
FROM points, elevation
WHERE ST_Intersects(rast, geom);
```



4) Proximity I

- **Determine the N geometries nearest from a set of other geometries**
- **N POINTs from 1 POINT (classical (wrong!) method)**

```
SELECT pointB.geom, pointB.id,
        ST_Distance(pointA.geom, pointB.geom) dist
FROM pointtableA pointA, pointtableB pointB
WHERE pointA.id = 999 AND ST_DWithin(pointA.geom, pointB.geom, 100)
ORDER BY dist
LIMIT 3;
```

Very hard, if not impossible to determine!

- **N POINTs from 1 POINT (KNN method with the <-> operator)**

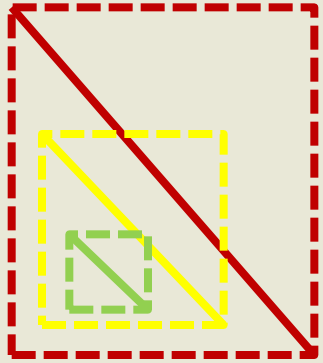
```
SELECT pointB.geom, pointB.id, ST_Distance(pointA.geom, pointB.geom) dist
FROM pointtableA pointA, pointtableB pointB
WHERE pointA.id = 999
ORDER BY (SELECT geom FROM pointtableA WHERE id = 999) <->
pointB.geom
LIMIT 3;
```

4) Proximity II

- **N POINTs for each POINTs of a table (KNN)**

```
SELECT pointA.id, pointA.geom, pointB.id id2,
        ST_Distance(pointA.geom, pointB.geom) dist
FROM pointtableA pointA, pointtableB pointB
WHERE pointB.id = ANY ((SELECT array(
                        SELECT id FROM pointtableB
                        ORDER BY geom <-> pointA.geom
                        LIMIT 3
                        ))::integer[])
ORDER BY pointA.id, dist;
```

- **N GEOMETRYs for each GEOMETRYs of a table (KNN)**
  - Problem: the <#> operator works only on bounding boxes
  - The bounding box of a further geometry could always be nearest than the nearest geometry.
  - Solution…



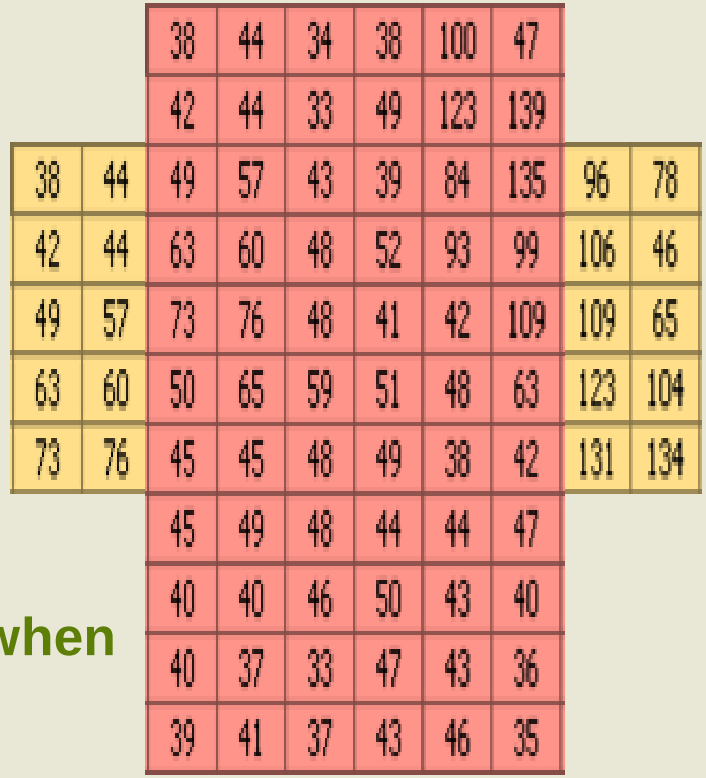
4) Proximity III

- N GEOMETRYs for each GEOMETRY of a table (KNN)
  - Select the 30 (or 100) nearest bounding boxes with KNN
  - Then sort by true nearest (distance)

```
WITH first30 AS (
  SELECT pointA.id, pointA.geom, pointB.id id2,
              ST_Distance(pointA.geom, pointB.geom) dist
  FROM pointtableA pointA, pointtableB pointB
  WHERE pointB.id = ANY ((SELECT array(
                                  SELECT id FROM pointtableB
                                  ORDER BY geom <#> pointA.geom
                                  LIMIT 30))::integer[])
  ORDER BY pointA.id, dist
), ordered AS (
SELECT *, ROW_NUMBER() OVER (PARTITION BY id ORDER BY dist) rownum
FROM first30
)
SELECT * FROM ordered WHERE rownum < 4;
```

5) Map Algebra I

- A very classical type of analysis on rasters
  - Output raster is the result of
    - a SQL expression
    - or a custom SQL fonction
    - evaluated for each pixel or the neighbour of each pixel of one or two input rasters.
  - The output raster extent can be equal to
    - the extent of the 1st raster,
    - the extent of the 2nd raster,
    - the intersection of both extents
    - or the union of both extents.
  - We can explicitely control what happens when one value is a nodata value.
  - Many functions a built over MapAlgebra
    - ST_Clip(rast, geom), ST_Intersection(rast, rast), ST_Union(rast)
    - ST_Aspect(), ST_Hillshade(), ST_Slope(), ST_Roughness()



5) Map Algebra II

- **Merge (union) rasters together from a raster time series**

Disjoints

```
SELECT year, ST_Union(rast) rast
FROM tiledrastseries
GROUP BY year;
```

Overlapping

```
SELECT year, ST_Union(rast, 'MEAN') rast
FROM tiledrastseries
GROUP BY ST_UpperLeftX(rast), ST_UpperLeftY(rast);
```

- **Compute hillshades for a tiled elevation raster coverage**

```
SELECT ST_HillShade(ST_Union(e2.rast), 1, e1.rast, '32BF', 180) rast
FROM elev e1, elev  e2
WHERE ST_Intersects(e1.rast, e2.rast)
GROUP BY e1.rast;
```

- **Add an elevation layer (tree tops) to normal elevation**

```
SELECT ST_MapAlgebra(e.rast, fc.rast, '[rast1] + [rast2]', NULL,'INTERSECTION')
FROM elevation e, forestcover fc
WHERE ST_Intersects(e.rast, fc.rast);
```

5) Map Algebra III

- **Reclassification of a 32BF raster into a 8BUI, 255 = nodata**

```
SELECT ST_SetBandNodataValue(ST_MapAlgebra(rast, '8BUI',
  'CASE
    WHEN 0 <= [rast] AND [rast] <= 150 THEN round(10 * [rast] / 150.0)
    WHEN 150 < [rast] AND [rast] <= 254 THEN 10 + round(10 * ([rast] - 150)/(254 - 150))
    ELSE 255
   END', 255), 255) rast
FROM hillshade;
```

- **Samething with ST_Reclass() (much faster)**

```
SELECT ST_Reclass(rast,
       ROW(1, '0-500:1-10, (500-10000:10-254', '8BUI', 255)::reclassarg) rast
FROM rasttable;
```

6) Rasterization of a vector coverage I With ST_AsRaster() & ST_MapAlgebra()

```
CREATE TABLE ramsafe_welltiled_forestcover_rast AS
WITH forestrast AS (
  SELECT rid, ST_MapAlgebra(
            ST_Union(ST_AsRaster(geom, rast, '32BF', height, -9999)),
            ST_AddBand(ST_MakeEmptyRaster(rast), '32BF'::text, -9999, -9999),
                              '[rast1]', '32BF', 'SECOND') rast
  FROM forestcover, elevation
  WHERE ST_Intersects(geom, rast)
  GROUP BY rid, rast
)
SELECT a.rid,
      CASE
        WHEN b.rid IS NULL THEN ST_AddBand(
                          ST_MakeEmptyRaster(a.rast), '32BF'::text, -9999, -9999)
        ELSE b.rast
      END rast
FROM elevation a LEFT OUTER JOIN forestrast b
ON a.rid = b.rid;
```

Fast but limited

- Only the value at pixel centroids can be extracted (GDAL)
- Only basic metrics (like the means of values) can be computed when many pixels overlaps (ST_Union)

6) Rasterization of a vector coverage II With Addons ST_ExtractToRaster()

```
CREATE INDEX forestcover_geom_gist ON forestcover USING gist (geom);

CREATE TABLE extracttoraster_forestcover AS
SELECT ST_ExtractToRaster(
                ST_AddBand(
                    ST_MakeEmptyRaster(rast), '32BF'::text, -9999, -9999),
                'public',
                'forestcover',
                'geom',
                'height',
                'MEAN_OF_VALUES_AT_PIXEL_CENTROID'
              ) rast
FROM elevationcoverage;
```

- **Many more methods!**
  - and easy to add more…

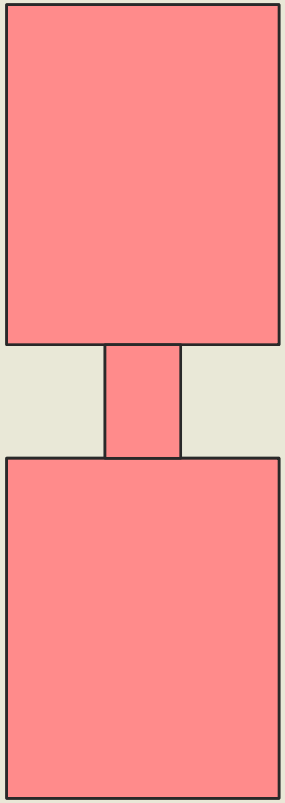
Vectorization is easy!

```
SELECT  (ST_DumpASPolygons(rast)).*
FROM rasttable
```

7) Overlap removal in a vector coverage

- **Strategy:** For each polygons, remove every parts overlapping with other polygons (using an aggregate function) and do it in a very ordonned way.

```
SELECT a.id, ST_DifferenceAgg(a.geom, b.geom) geom
FROM overlappingtable a,
        overlappingtable b
WHERE ST_Equals(a.geom, b.geom) OR
     ((ST_Contains(a.geom, b.geom) OR
       ST_Contains(b.geom, a.geom) OR
       ST_Overlaps(a.geom, b.geom)) AND
      (ST_Area(a.geom) < ST_Area(b.geom) OR
       (ST_Area(a.geom) = ST_Area(b.geom) AND
        ST_AsText(a.geom) < ST_AsText(b.geom))))
GROUP BY a.id;
```

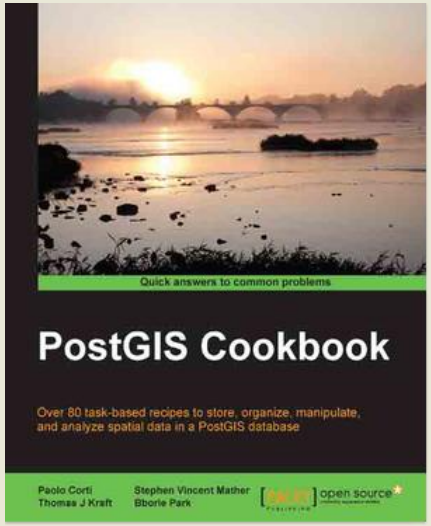


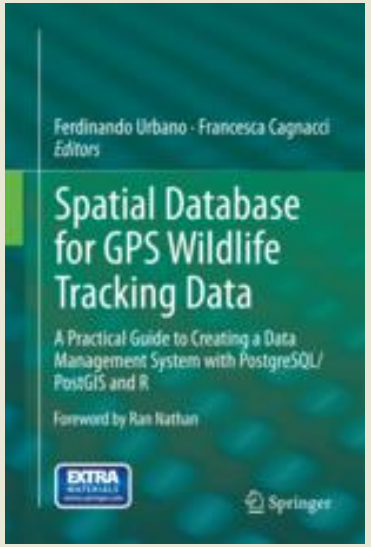
- ST_DifferenceAgg() is in the PostGIS Addons

What's missing for raster in PostGIS?

- PostGIS functions
  - Interpolations methods - to build continuous raster coverages from point coverages (lidar –> raster)
  - Cost analysis and shortest path rom a cost raster, not a vector network (pgRouting)
  - Viewshed
- pgsql2raster and a GUI interface to the loader and dumper
- Better integration of PostGIS raster in QGIS
  - Load PostGIS rasters from the "Add PostGIS Layer" dialog (not only from the DB Manager)
  - Load one table not as a big raster but as a coverage, possibly irregular
  - Make sure we can symbolize this raster coverage as a whole coverage

Ressources





- PostGIS documentation
- Online tutorials
- postgis-users discussion group
- GIS Stack Exchange
- PostGIS Planet
- Geospatial Elucubrations

Thanks!

http://trac.osgeo.org/postgis/wiki/WKTRaster

Boreal Avian Modelling Project

The Canadian BEACONs Project
Boreal Ecosystems Analysis of Conservation Networks

western boreal conservation initiative
Environment Canada
l'initiative de conservation boréale de l'ouest
Environnement Canada

Fondation canadienne pour l'innovation
Canada Foundation for Innovation

PostgreSQL

PostGIS
Geospatial Objects for PostgreSQL

Cadcorp
WORLD LEADING GIS SOFTWARE

deimos

PARAGON CORPORATION

azavea

MichiganTech
Create the Future

cef
Centre d'étude de la forêt

UNIVERSITÉ LAVAL

España Virtual


Bborie


Jorge


Pierre


Regina


Mateusz


Sandro


David