Advanced Spatial Analysis with PostGIS

UNIVERSITÉ LAVAL

cef
Centre d'étude de la forêt

Pierre Racine
Research assistant
GeoElucubrations

What are we going to do?

1) GeoTable Summary
2) Overlap Removal (4 methods)
3) Gap Filling
4) Vector/Vector Analyses
   - Extraction from a polygon coverage for points
   - Extraction from a polygon coverage for polygons
5) Vector/Raster Analyses
   - Extraction from a raster coverage for points
   - Extraction from a raster coverage for polygons
6) Elevation Profiles
7) Proximity Analyses
   - N nearest points from one point
   - N nearest geometries for each geometry of a table
8) Raster/Raster Analyses
   - Map algebra
   - Union of overlapping rasters
9) Rasterization of Vector Coverages

1 hour

1 hour

1 hour

1 hour

PostGIS Add-ons

- **A single SQL file of PL/pgSQL functions**
  - With tests and uninstall script
  - Goal: Make it easy to share users contributed PostGIS functions

Have a look at Geospatial Elucubrations!

- **Most notably:**
  - **ST_GeoTableSummary()** provides a summary of the topological characteristics of the table.
  - **ST_DifferenceAgg()** and **ST_SplitAgg()** to remove overlaps in a polygon coverage.
  - **ST_ExtractToRaster()** to extract any metric from a vector coverage into a raster. e.g. nb. of points, value of biggest polygon, and more… 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_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.

1) GeoTable Summary

- **ST_GeoTableSummary(**
  schema, table, geom, id,
  nbhistobins,
  list_of_summaru_to_do, list_of_summaru_to_skip,
  where_clause
)
- **9 types of summary**
  1. **Duplicate ids (s1 or iddup)**
  2. **Duplicate geometries (s2, gdup or geodup)**
  3. **Overlapping geometries (s3 or ovl)**
  4. **Geometry types (s4, types, gtypes or geotypes)**
  5. **Vertexes stats (min, max, mean) (s5 or vertx)**
  6. **Vertexes histogram (s6 or vhisto)**
  7. **Areas stats (min, max, mean) (s7, area or areas)**
  8. **Areas histogram (s8 or ahisto)**
  9. **Small areas count (s9 or sacount)**
- **Typical statement**

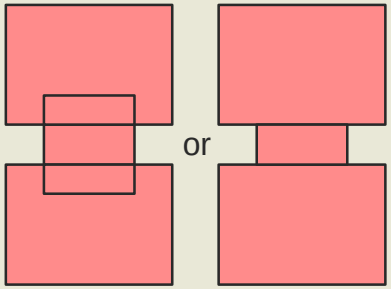
SELECT * FROM ST_GeoTableSummary('public', 'geotable', 'geom', 'id', 10, 'gtypes')

Still a lot of work to do:

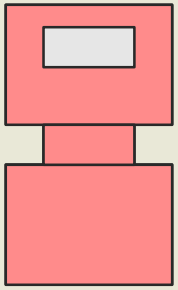
- Add gap summary
- Add a fixquery column
- Support tables of linestrings
  - Intersections instead of overlaps
  - Length instead of areas
- Make it an aggregate?

2) Three methods to remove overlaps

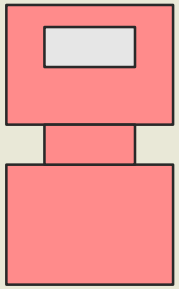
1) Ring method
Extract exterior rings of each polygon, union and polygonize them



- Perfect: no overlaps remains
- Quite slow
- ST_Union() might generate memory overflow
- Remove all holes
- Gaps are merged together

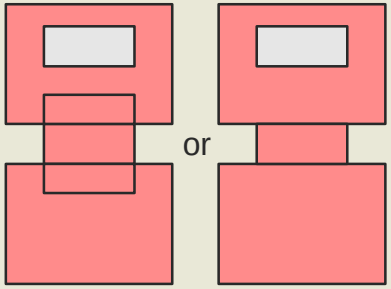


2) Clip method
Clip each polygon with an aggregate of all overlapping polygons



- Very fast
- Imperfect: some tiny overlaps remains
- Depends on ST_differenceAgg() in the Add-ons

3) Split method
Cut each polygon with an aggregate of all overlapping ones and remove duplicates



- Fast
- Imperfect: some tiny overlaps remains
- Depends on ST_SplitAgg() in the Add-ons

2.1) Exterior Rings Method

1) Extract all the polygons exterior rings as simple polygons with ST_ExteriorRing() and ST_MakePolygon() into a new table (holes are lost)

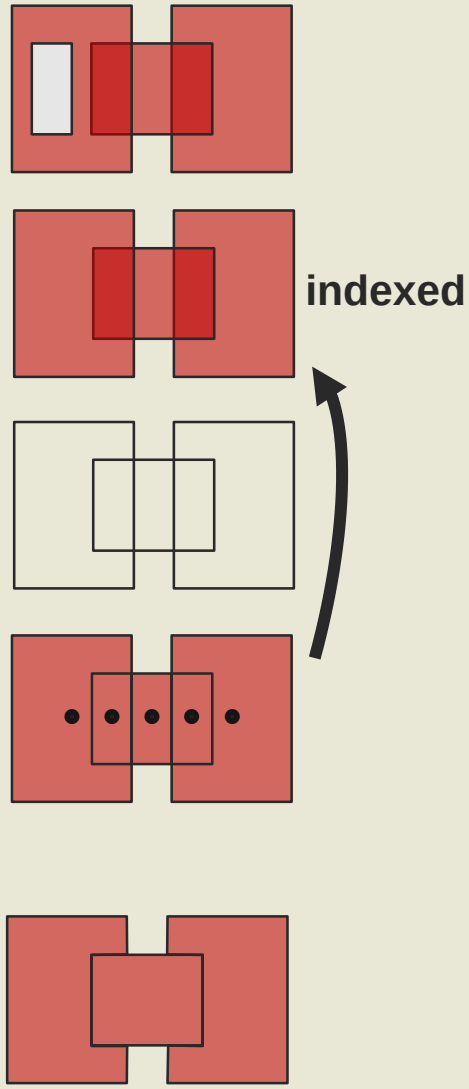
2) Index this table (for later)

3) Union all the polygons exterior rings together into a single geometry with ST_Union() and ST_ExteriorRing()
(attributes are lost)

4) Reconstruct polygons from these rings ST_Polygonize() and ST_Dump()
- Multi-polygons broken into many polygons. Re-union them later.
- Gaps are transformed into polygons. Delete them later.
- No more overlaps from this point…

5) Left join each DISTINCT (sorted by id DESC) polygon centroid back with the original polygon to get the right id ST_PointOnsurface() and ST_Within()

6) Re-union polygons sharing the same id together

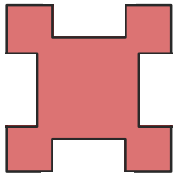


2.2 and 2.3) Difference and Split Aggregate Methods

- **What is an aggregate function?** e.g. avg()
- Three functions
  1. **State function** aggregates all the values into a state variable (sum and count). This variable can be complex: ARRAY or TYPE
  2. **Final function** determine the final value (sum/count)
  3. **Aggregate function** links the state and final functions into a single function
- Groups of aggregated values are determined by GROUP BY

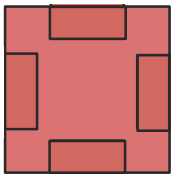
The difference aggregate method

- **ST_DifferenceAgg(geomA, geomB)**
- The state function removes (using ST_Difference()) all **geomB** from **geomA** (except the first **geomB** identical to **geomA**)
- The final function simply returns the clipped geometry



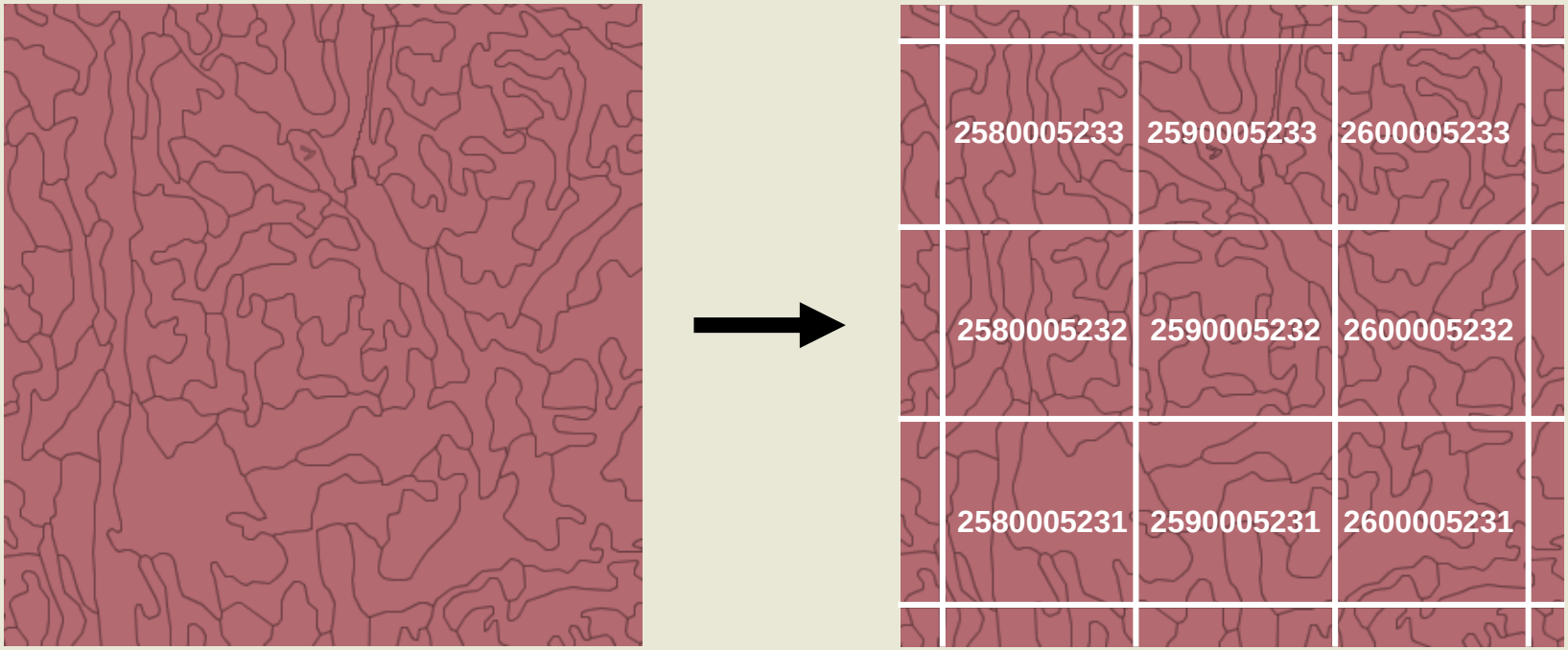
The split aggregate method

- **ST_SplitAgg(geomA, geomB)**
- The state function split (using ST_Difference()) **geomA** with all **geomB**
- The final function returns an array of the splitted geometries



3.4) Parallelizing Overlap Removal by Splitting the Coverage as a Grid

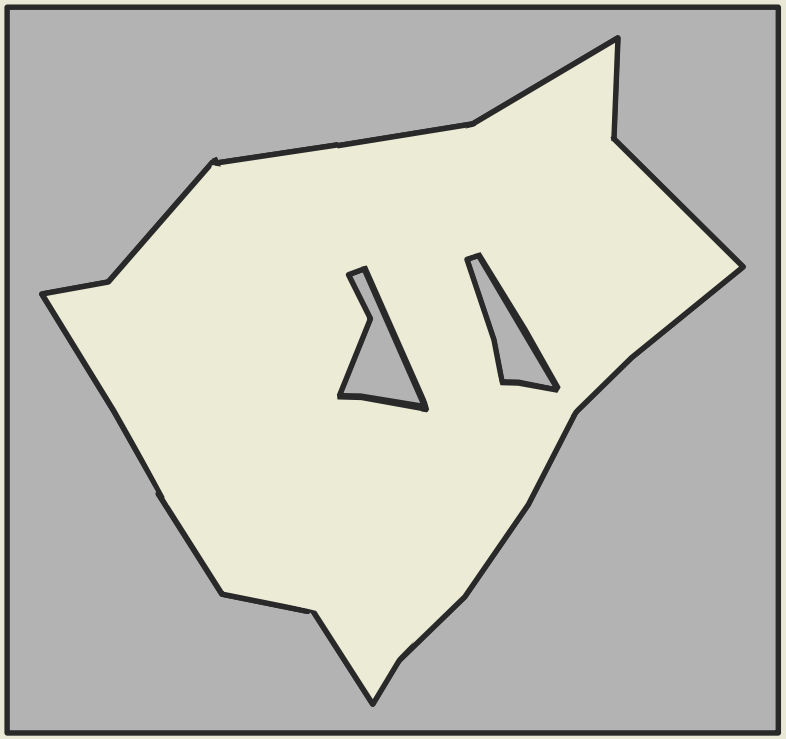
- (ST_SplitByGrid(geom, 1000)).* returns
  - each polygon splitted by a global grid and
  - the unique identifier of the cell (bigint)
  - the x and the y



- Each polygon has to be reunioned afterward (ST_Union)

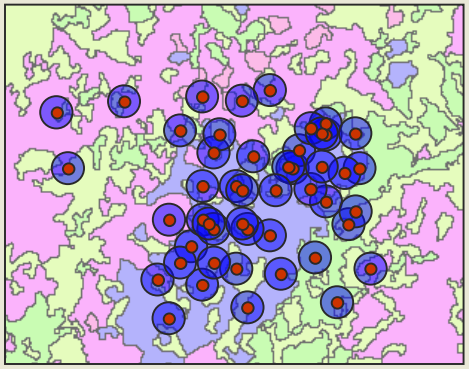
3) Gap Filling

1. Create a polygon covering all the other polygons with ST_Extent() and ST_Buffer().
2. Remove the union of all the inner polygons from the outer polygon with ST_Union() and ST_Difference().
3. Dump the remaining polygon into its many parts with ST_Dump().
4. Drop the first polygon part (the biggest) with a WHERE clause.
5. Union the remaining parts with the biggest neighbor with ST_Union().



4) 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
…	…

=

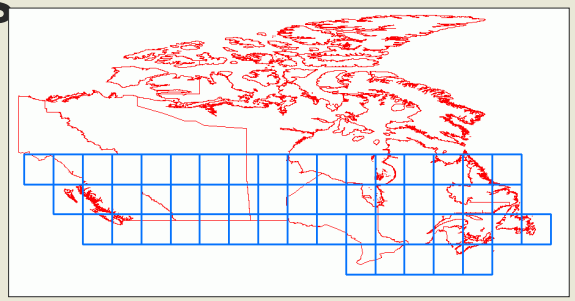
result			
geom	obsid	ctype	area
polygon	24	4	10.34
polygon	53	3	11.23
polygon	24	5	14.23
polygon	23	2	9.45
…	…	…	…

```
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



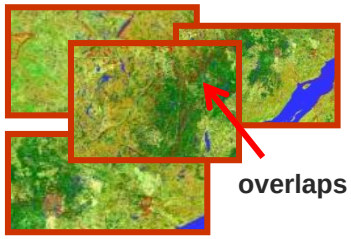
Why storing raster in the database?

- 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.
- One simple store for everything
  - Your vector data are normally already in the database…
- Performance
  - Analysis processing is normally faster on tiled raster coverages.
- 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

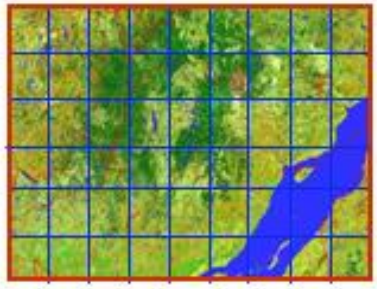
Rasters vs Tiles

- 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…)

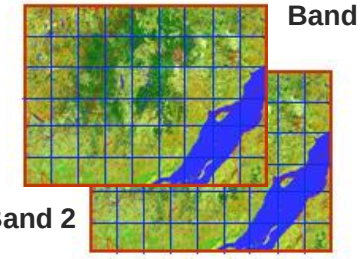
Possibles RASTER Arrangements



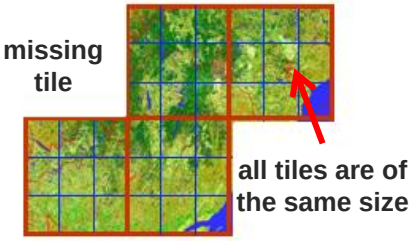
a) Satellite or aerial imagery warehouseou (4 images, 4 rows)



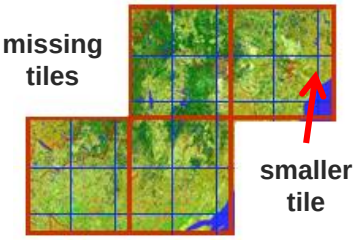
b) Regularly tiled rectangular raster coverage (54 tiles, 54 rows)



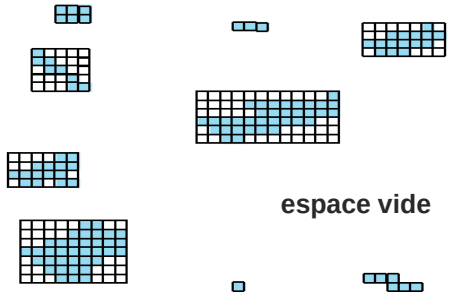
c) Multi-band tiled images (1 table of 108 tiles)



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



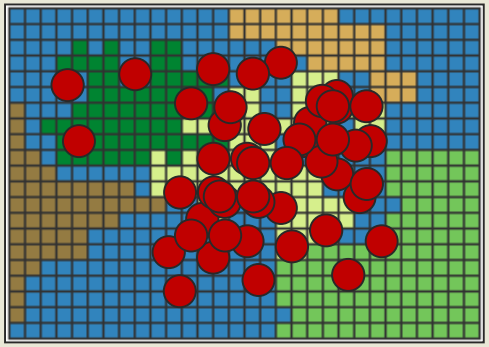
e) Irregularly tiled raster coverage (36 tiles, 36 rows)



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

5) 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



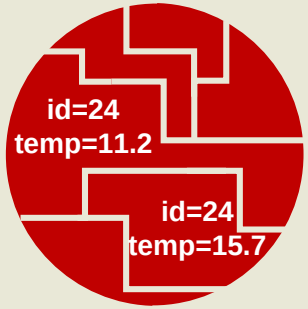
buffers	
geom	pointid
polygon	24
polygon	46
polygon	31
polygon	45
…	…

∩

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;
```

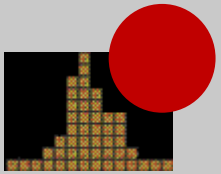
- when polygons size relatively smaller than pixel size
- or vector coverage is composed of lines
- slower, more precise

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;
```

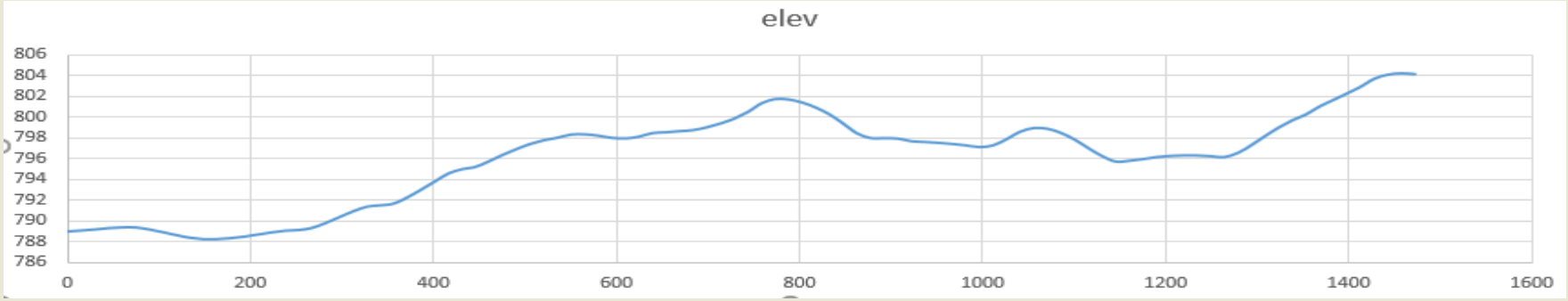
- when polygon size much bigger than pixel size
- works only with polygons coverage
- faster, less precise

Intersects ignore nodata values



6) Elevation Profiles

```
WITH road AS (
  SELECT (ST_Dump(geom)).geom
  FROM roads_table WHERE id = 1806
), points AS (
  SELECT id,
         round((id * ST_Length(geom))/99, 1) length,
         ST_LineInterpolatePoint(geom, id/99.0) geom
  FROM generate_series(0, 99) id, road
)
SELECT id, geom, length,
       ST_Value(rast, geom) elev
FROM points, elevation_table
WHERE ST_Intersects(rast, geom);
```



7) 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 id, geom,
        (SELECT geom
         FROM pointtableA WHERE id = 146) <-> pointB.geom dist
FROM pointtableB pointB
ORDER BY dist
LIMIT 3;
```

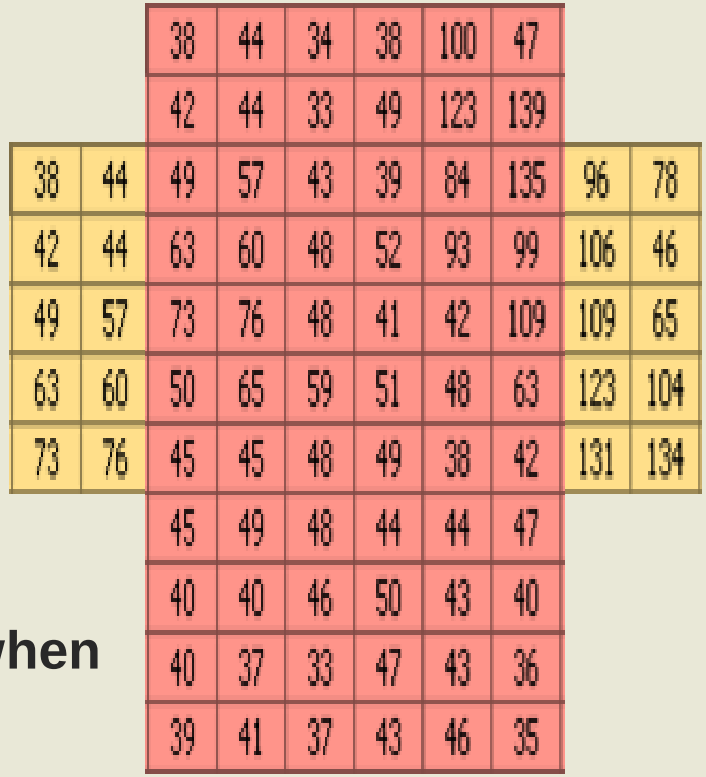
7) Proximity II

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

```
SELECT geomA.id,
            geomA.geom,
            near_geom.id near_id,
            near_geom.geom near_geom,
            near_geom.dist
FROM geomtableA geomA, LATERAL
          (SELECT geomB.id,
                       geomB.geom,
                       geomB.geom <-> geomA.geom dist
           FROM geomtableB geomB
           ORDER BY dist
           LIMIT 3) near_geom
ORDER BY geomA.id, near_geom.dist;
```

8) Map Algebra I

- A very classical type of analysis on rasters
- Output raster is the result of
  - a SQL expression
  - or a custom user written 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()



8) 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);
```

8) 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;
```

9) Rasterization of Vector Coverages I
ST_AsRaster() & ST_MapAlgebra() Method

```
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', NULL, '[rast1]') 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)

9) Rasterization of Vector Coverages II
Add-ons ST_ExtractToRaster() Method

```
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…

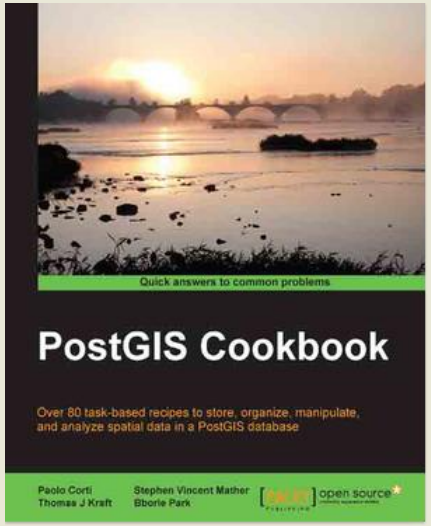
Vectorization is easy!

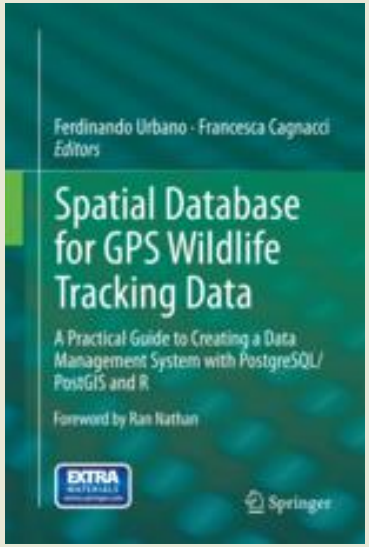
```
SELECT  (ST_DumpASPolygons(rast)).*
FROM rasttable
```

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
- Planet PostGIS
- 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