POSTGIS: A QUICK INTRO

W. Webb Sprague
2007-10-20
PDX Postgresql Day

PRESENTATION OVERVIEW

- Sections:
  - Geographic modeling -- how many geography folks in the room?
  - GIS/DB overview
  - Getting started
  - PostGIS geographic functions, types, etc
  - Extended example
  - Closing thoughts
- Can only give a taste in this presentation

WHO AM I?

- 4th year Ph D in Demography UC Berkeley (“counting people”)
- An interest in cultural influences on demographic processes
- Doing field research in Oregon
- Employed as scientific and database programmer

ACKNOWLEDGMENTS

- All the hackers that make Postgres happen
- Refractions
- Selena and PDXPUG
- Oregon Spatial Data: http://www.oregon.gov/DAS/EISPD/GEO/alphalist.shtml
- Portland State
- Sponsors

GEOGRAPHY CONCEPTS

Geographic modeling for database developers and administrators

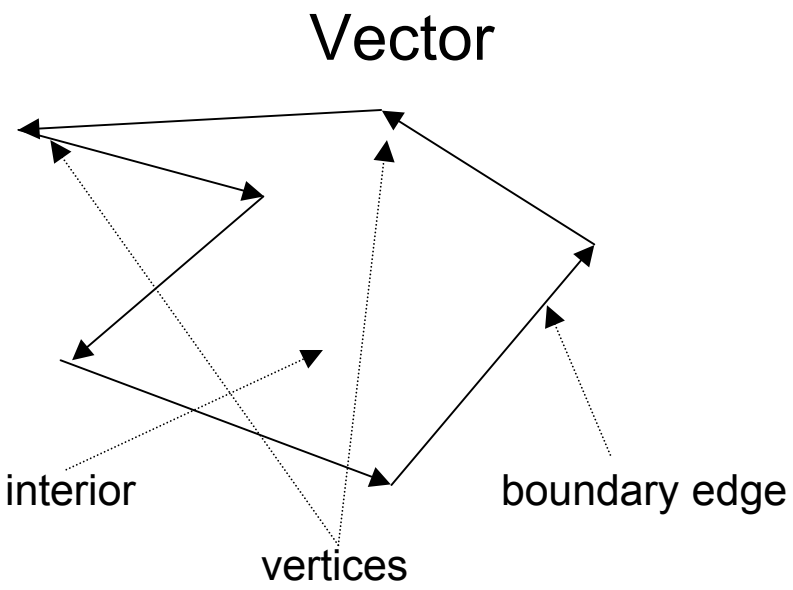
Cartography versus geographic data

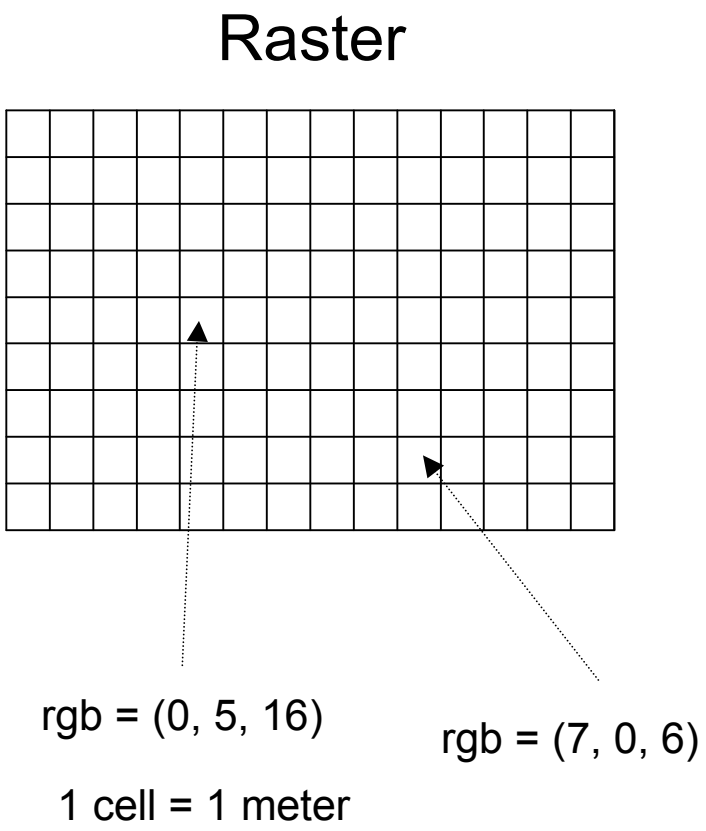
- Geography doesn't need maps
  - Presentation <> storage
  - Geographic information interesting in its own right without maps - cancer rates by proximity to toxics, lists of sales-leads by zip, etc
  - Cartography a different art than database management and development
  - Only bad cartography in this presentation…

Two types of geographic models

- Vectors
  - a single arbitrary shape
  - stored as a set of ordered vertices
  - there are “multi” vectors
  - only model we will use here
- Rasters
  - a homogeneous grid of cells
  - each cell has a vector of information (“bands”)
  - each pixel corresponds to a resolution - e.g. 1 pixel = 1 meter
- Others - “DEM”, “TIN”, etc.

Geographic models picture





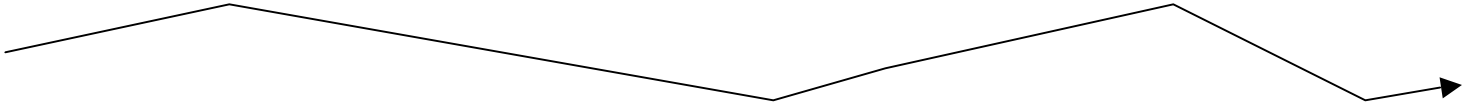
Vector types by dimension

- **Points** - 0 dimension, single vertex. Town locations, road intersections, etc
- **Lines** - 1 dimension, multiple vertices, oriented, no interior/ exterior, never rejoins itself. Roads, rivers, etc
- **Polygons** - 2 dimensions, multiple vertices, oriented (counter clockwise?), divided into interior/ exterior, rejoins itself to make closed shape (unless invalid). Physical regions, political boundaries, etc
- **Bounding Boxes** - the square polygon that minimally encompasses another shape. Only two corners. Used in indexes.

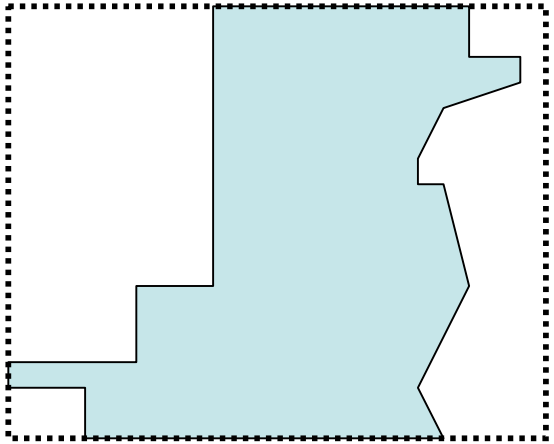
Vector types picture

Point: “Halsey Oregon. Lat=XX, Long=XX.”

Line: Highway 20.



Polygon and bounding box: Benton County.



Levels of detail/ abstraction

- The map scale changes how much detail is used to describe a feature:
  - A road might be modeled with 5 or 500 vertices depending on the scale.
- The map scale also changes how dimensions are used to model a feature:
  - A short road might be modeled as a point if we are mapping North America, …
  - … a line in a map of Oregon, …
  - … or a polygon in a hydrology map of a subdivision.

Geographic and tabular data

- There are two parallel datasets for each vector:
  - The geography, which describes each vertex, the orientation, etc.
  - the data, which has an id and multiple columns.
- (These two parallel data sets are found in one row PostGIS)

Layers

- Multiple layers in geographic models
  - Typically there is a different layer for each entity - counties, roads, towns, etc
  - Overlay possible because of common projection
  - Mylar and colored pencils in the old days.

Projections

- Every geographic dataset needs to have a “projection”, a coordinate reference that fixes it on the planet.
- Especially important for overlay analysis
- Vectors fix each vertex against this reference.
- Rasters fix two corners and the pixel size (I think, but might use a corner and the orientation of the matrix).
- There is a difference between projection onto a plane and lat/long coordinates.
- SRID - Spatial Referencing system Identifier

Imputing geography

- Sometimes we have data that isn't geographic - say sale instances by zip code - but we want to examine its geographic characteristics:
- use a key in the non-geographic table to join to a table with geography - like a zip code table with polygons of zip code region,
- impute the geography of the sales instance by centroid() or pointOnSurface(),
- make geographic queries with these imputed points.
- This is especially important if adding geography to an existing database.

Two important types of geo storage

- Shapefiles - many files in a directory
  - ".shp" stores geography
  - ".dbf" stores associated data
  - ".prj" stores projection
  - ".idx" is some sort of index
  - Always ship these as zip-files to keep everything in a single directory
- Tiff's
  - Big grids with (generally) floating point numbers for each cell
  - "geotiffs" also have a file for projection and scale

Other geo storage

- Others: Autocad, MapInfo, Maptitude, etc
- Remote sensing rasters - many more bands than used for visible color. Like Tiff's but different.
- Oracle?
- ESRI Geodatabases
  - Personal geodatabase in MS Access -- yuck!
  - Some other SQL Server thing

GIS / DB OVERVIEW

History of DBs

- For the next presentation…

Open source DBs

- Really there is only one that matters,…

History of GIS

- John Snow and London Cholera
- Florence Nightingale
- Mylar and colored pencils
- 1962 CGIS
- 1980s ESRI, MapInfo, etc
- 1982 GRASS US Army
- 1990s GIS takes off
- 2005-01-13 Postgis RC1 released

Other important Open source GIS projects

- Mapserver
  - Mapscript a language for generating maps
  - Server infrastructure for displaying on www
- GDAL - library for processing raster data
- proj - library for re-projecting data
- GRASS - raster based, kind of clunky.
- Udig, QGIS, JUMP, Open EV, etc - display and editing
- GMT

Overview of PostGIS

- History - don’t know
- Approach: give each feature its own row with both geometry as just another column
- Excellent marriage of SQL and geography, though requires familiarity with both

GETTING STARTED

Installing postgres

- I think we all have managed to install postgres….

Installing postgis on host

- Need the “proj” and “geos” libraries
- configure --prefix=/$PGIS && make && make install
- This installs binary libraries and some SQL in /postgis. This does NOT make a database geographic -- at least not yet…

Setting up postgis for a db

- createdb foo_gis
- && createlang foo_gis plpgsql
- && psql foo_gis -f $PGIS/lwpostgis.sql
- && psql -f $PGIS/spatial_ref_sys.sql

PostGIS support tables

- “Geometry_columns”
  - an entry for each geometry column
  - used in reprojecting and by applications
- “Spatial_ref_sys”
  - a list of all possible projections
  - conversion scripts for proj

PostGIS schema

- The lw_postgis.sql file dumps everything in the public schema, causing icky namespace pollution.
- Solution:
  - CREAT SCHEMA postgis,
  - edit lw_postgis.sql adding “set search_path=postgis,public” at top,
  - remember to set search_path before trying to use PostGIS functions
- Problem - have to set your search_path later

Dumping and recreating

- Weird errors?
- Need to re-enable a geographic database as in previous slide before un-dumping a pg_dump

Postgis docs

- http://postgis.refractions.net/docs/
- http://postgis.refractions.net/docs/ch04.html#id2908428 - examples
- Great docs (yay Refractions)

DOING GEOGRAPHY IN POSTGIS

An overview and an assortment of useful functions, types, etc

Postgis datatypes

- “geometry” datatype
- a big glob of binary stuff, including data, projection, other magic
- also “box” datatype
- use functions to access
- can easily convert to and from text

Geo function overview, part I

- Returning booleans - e.g. “within(geo1, geo2)”.
- Returning transformed scalar geometries - “centroid(geo)”
- Returning aggregated geometries - “geomunion(geo1, geo2)”
- Geographic editing - “addpoint(geo1, geo2)”

Geo function overview, part II

- Data validation - “isValid()”
- Character of geometry - “numpoints()”, “setsrid()”
- Administration - “(add|drop)geometrycolumn()”

Geographic “relate()”

- Describes the overlap of two geometries, underlying above functions
- Dimension of overlap
- 3 by 3 matrix
- Weird, but the underlying function of all the booleans…

Geography from text

- select ST_GeomFromEWKT(
  'POLYGON(
  (0 0,4 0,4 4,0 4,0 0),
  (1 1, 2 1, 2 2, 1 2,1 1))'
  ));
- select asewkt(the_geom)
- Note that this means we can easily insert geometries represented as text coordinates.

Indexes

- CREATE INDEX "puma_boundaries_the_geom_gist" ON "puma_boundaries" using gist ("the_geom" gist_geometry_ops);
- Use the “ST_” functions to take advantage of indexes

EXTENDED EXAMPLE

Importing geographic data

- Import a shapefile (counties table used in example)
- shp2pgsql -s 2992 -D -i -I orcnty24 counties > counties.sql
- psql gis_db -f counties.sql
- If you have to re-project, let me know and I will send you a script…

Importing non geographic data

- (IPUMS ACS data)

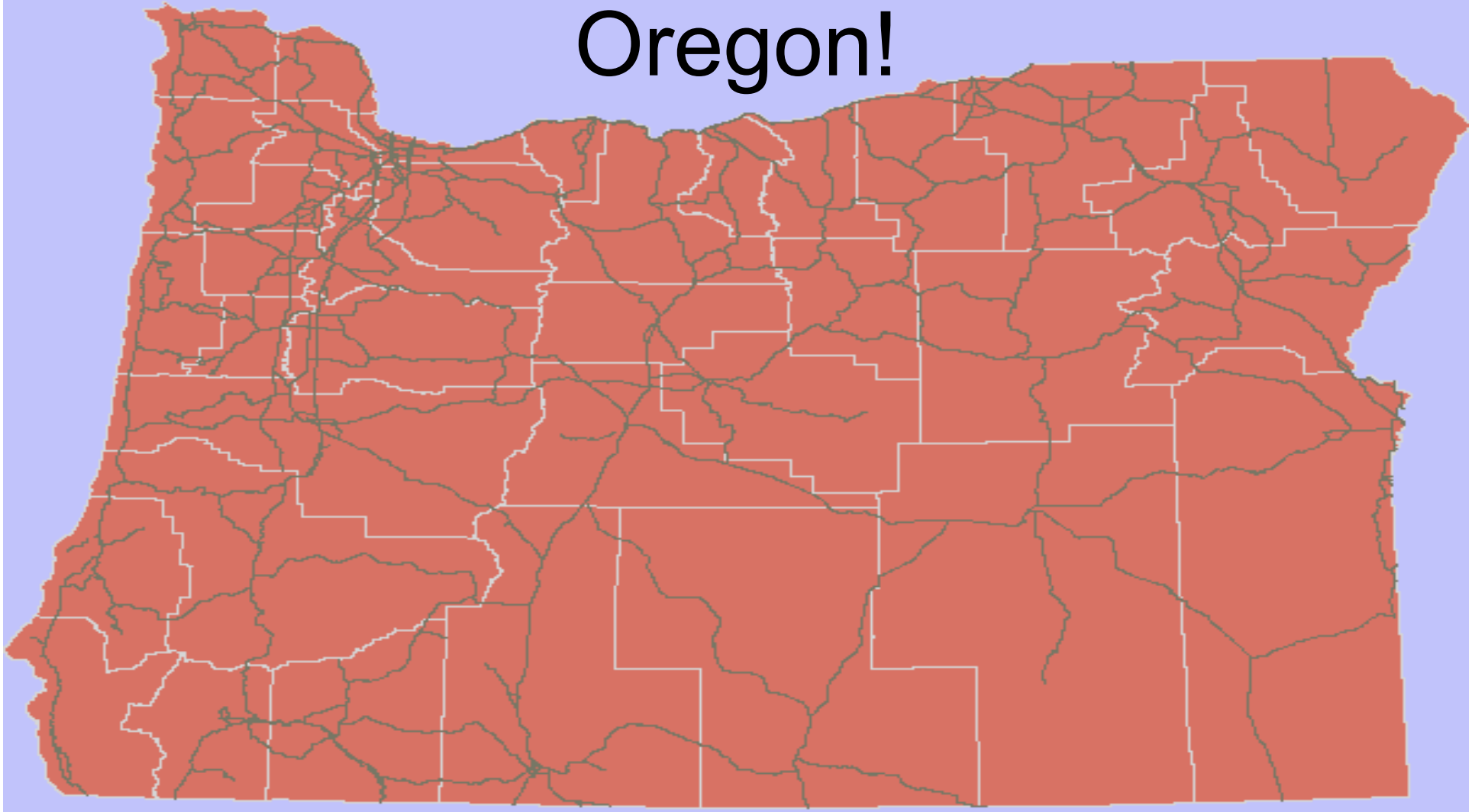
Description of SQL

I think we all know SQL…

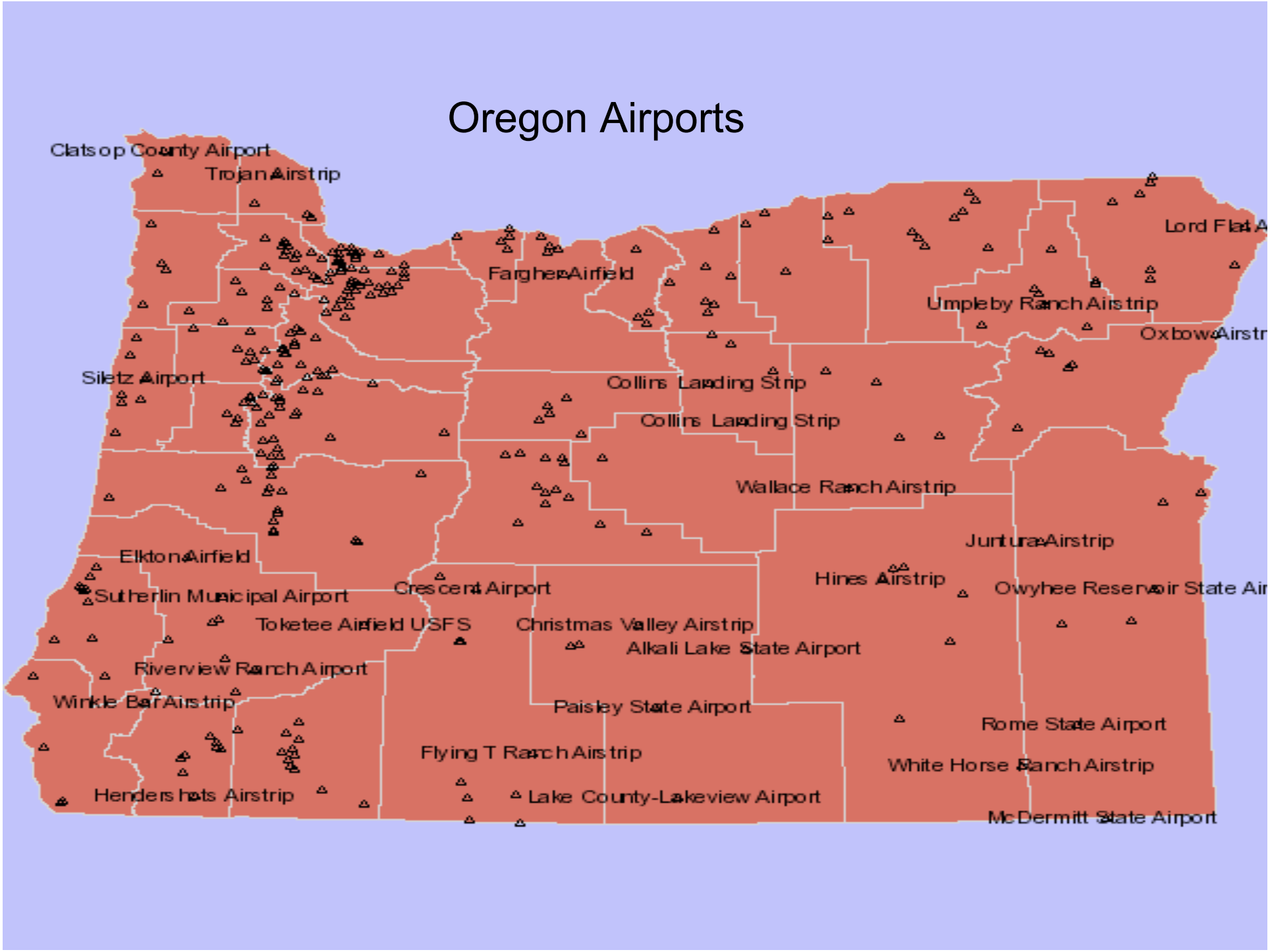
Example Overview

- Data tables: Placenames, City Limits, Highways, Counties
- My code: “postgis_code.sql”
- Results: “psql or_gis -a -f postgis_code.sql > sql_results.txt”

Oregon!



Oregon Airports

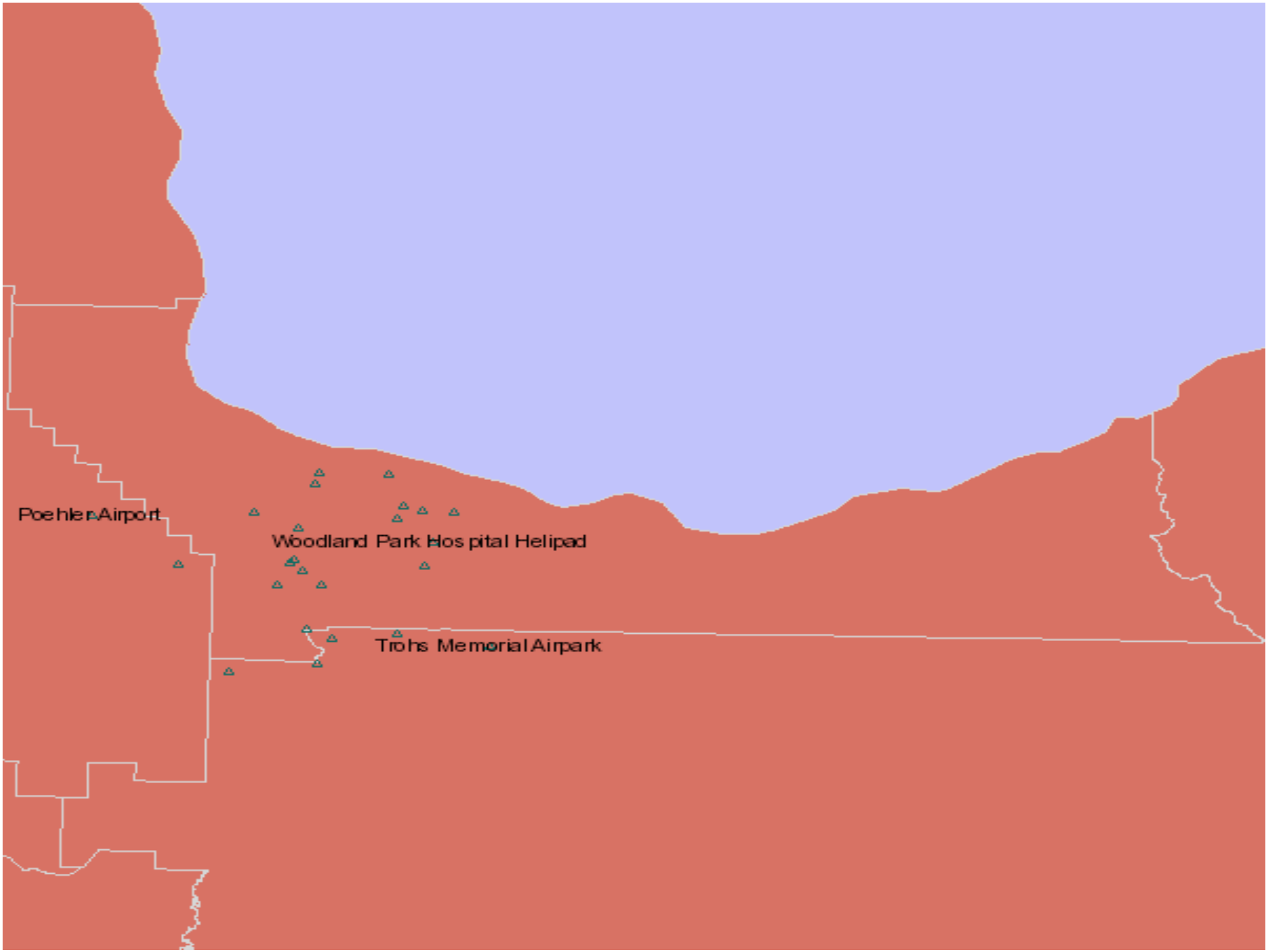


Examples I - fun with points

- select all airports and containing cities (Ex 1)

select count (>2) of airports by city limits (Ex 2)

select airports within 10 miles of Portland (Ex 3)

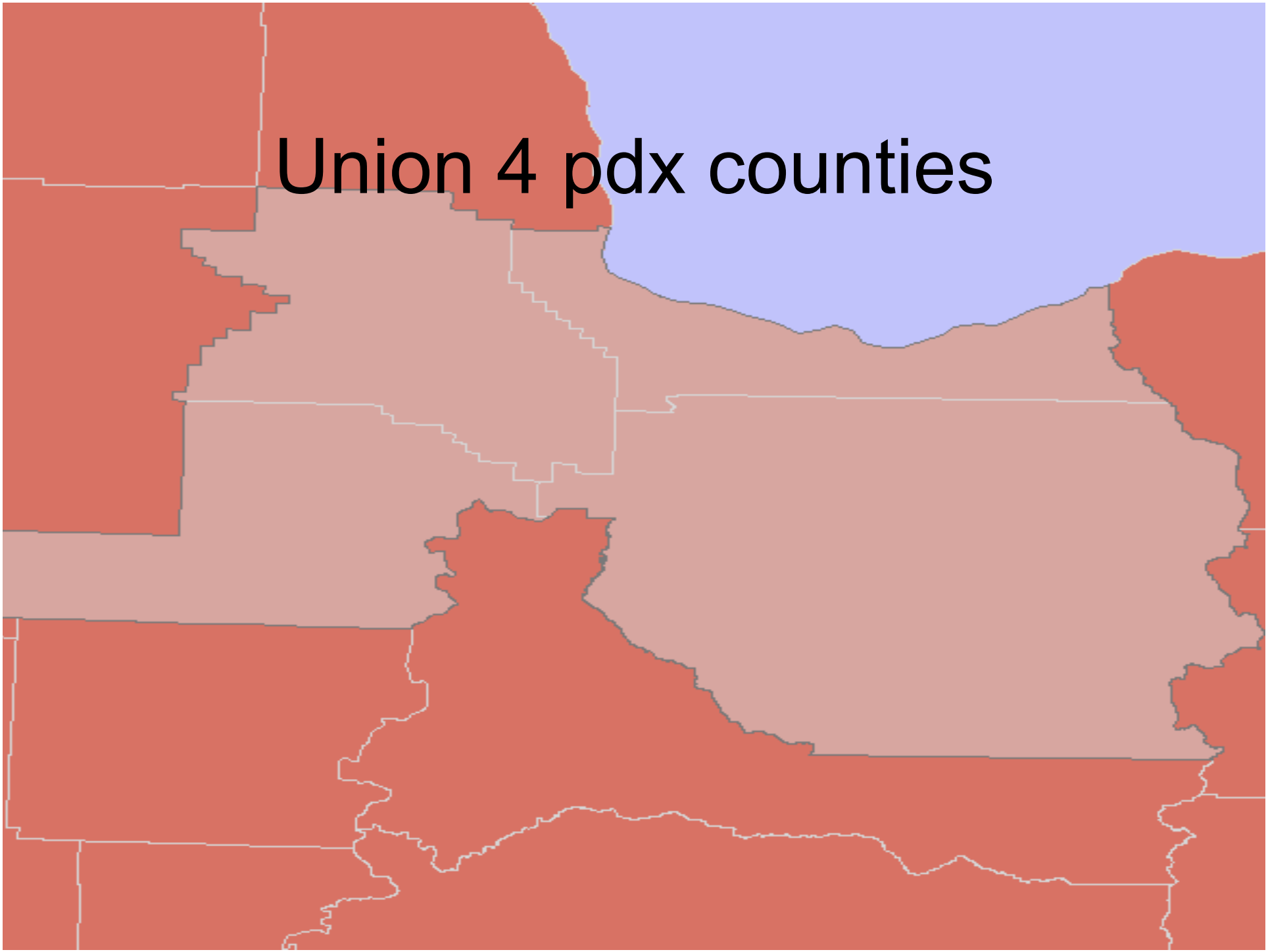

Poehler Airport
Woodland Park Hospital Helipad
Trohs Memorial Airpark

Examples II - select geographies

- select cities based on their area (Ex 4)
- select neighboring counties the four PDX metro counties (Ex 5)

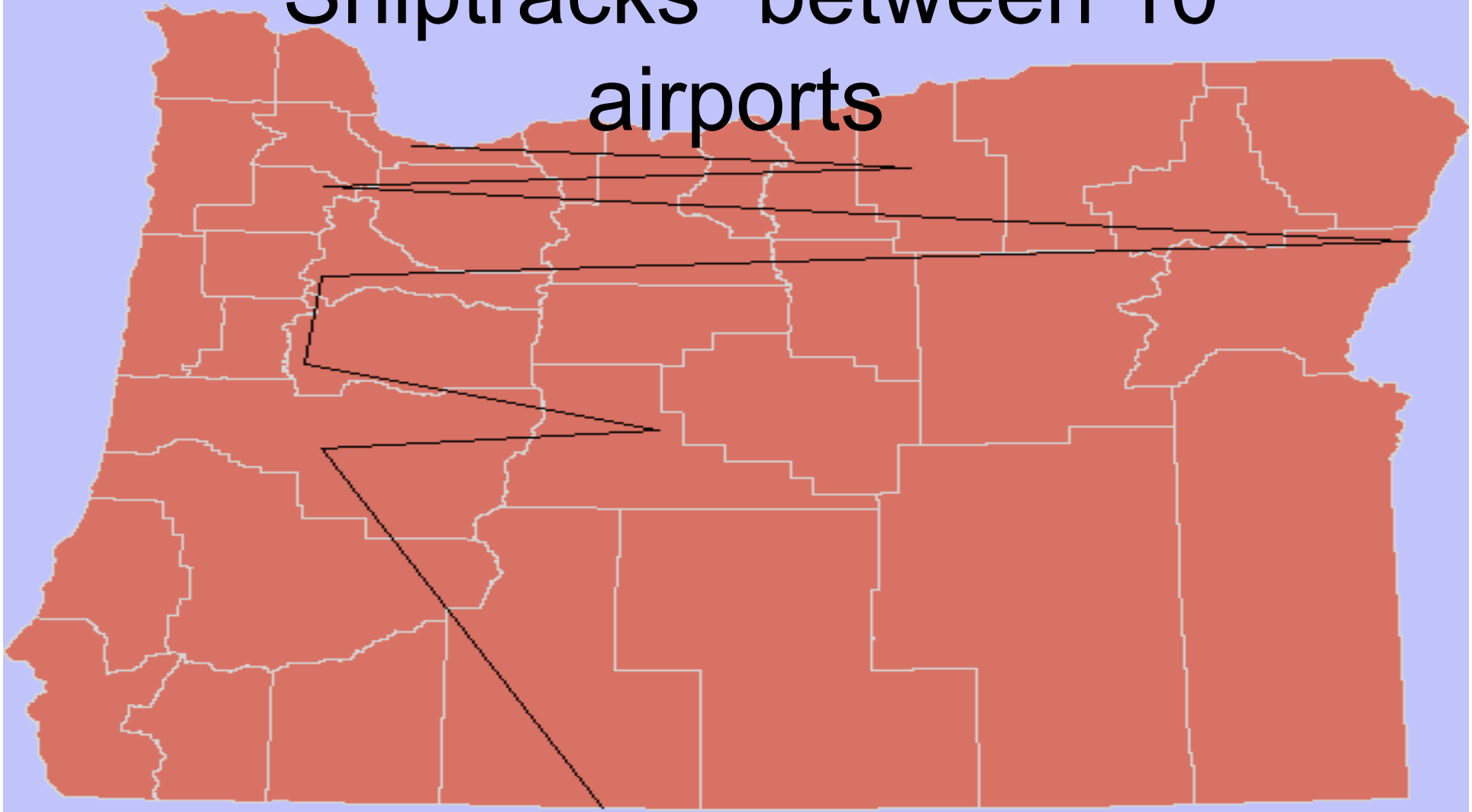
Examples III - aggregate-transform geometries

Collect the four PDX metro counties (Ex 6)


Union 4 pdx counties

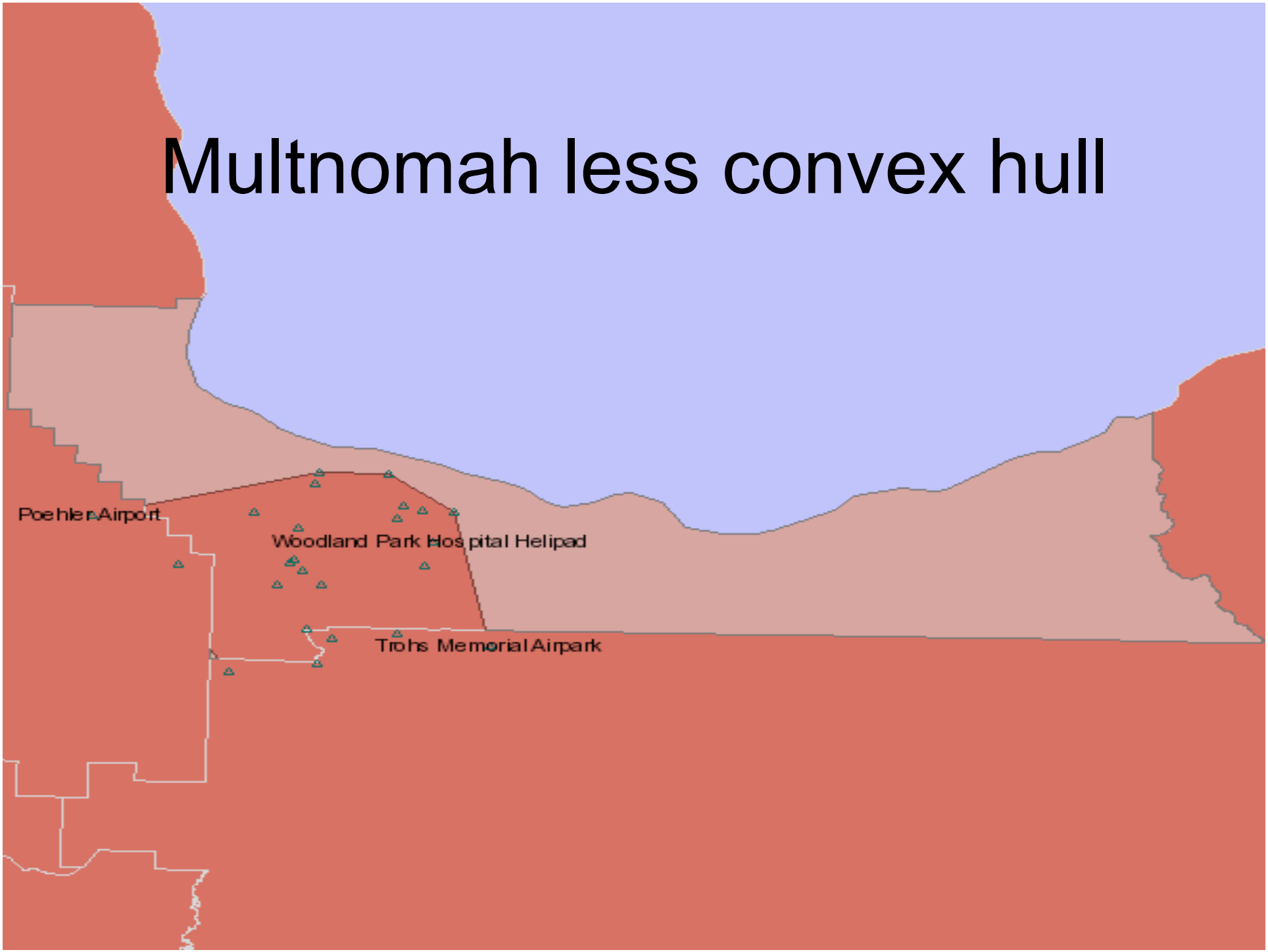
Create “shiptracks” from points (Ex 7)

“Shiptracks” between 10 airports



Geometric difference of convex hull formed by nearby airports and multnomah county (Ex 8)

Multnomah less convex hull



Fun with highways (Ex 9)

Quick chat - scalar-transform geographies

- centroid
- simplify
- de-normalizing geograhies

Displaying data

- Mapserver with above data - I used shp2img
- GMT
- QGIS, openEV, uDig etc
- Proprietary software starting to have support for postgis
- pgsql2shp -> ESRI -> fancy cartography

CLOSING THOUGHTS

Killer Application?

Further reading

- http://www.bostongis.com/
- http://www.spatialreference.org/
- http://www.maptools.org/

Further considerations

- Web-based GIS
- Digitizing and editing
- Geocoding
- Routing and networks

Creative possibilities of PostGIS

- Geographically enabling databases already (e.g. mailing lists)
- Streaming GPS data and triggers -- Shiptracks, etc
- Google map mashups - a python class with tablename inputs, inheriting from a Cheetah template to make the javascript?
- Others? Discussion?