Introduction to Postgres and PostGIS

Databases: what are they?

- Usually when you hear the word “database” a person is referring to “Relational Databases” or “Relational Database Management Systems” (RDBMS)

- For the purposes of this talk, a database is: “**A system that allows for the creation and manipulation of tables of data**”

- Data in our tables can be references to rows in other tables (the “Relations” in our “Relational Database”)

Database Example: Hospitals

hospital

id	name	telephone	general_beds	special_care_beds	address
1	SunnyTown Hospital	123-555-1234	500	150	1234 Main Street, Townsville, Washington
2	Lady of the Lake	123-555-2345	600	200	555 Street Rd, Cityville, Washington
3	Charity Hospital	504-555-1234	2600	1000	1532 Tulane Ave, New Orleans, Louisiana
4	UW Medical Center	206-123-4567	450	100	1959 NE. Pacific Street, Seattle, Washington

department

hospital_id	department	employees
2	Gastroenterology	10
2	Neurology	5
2	Primary Care	50
3	Primary Care	40
3	Rheumatology	5
4	Primary Care	45
4	Rehabilitation Medicine	15

Structured Query Language: SQL Examples

```
SELECT *
FROM hospitals
WHERE beds > 1000
```

3	Charity Hospital	504-555-1234	2600	1000	1532 Tulane Ave, New Orleans, Louisiana

```
SELECT d.department
FROM hospitals h
INNER JOIN departments d ON h.id = d.hospital_id
WHERE h.name = 'Lady of the Lake'
```

Gastroenterology
Neurology
Primary Care

So where does GIS come in?

- Our simple database allows us to answer questions about Hospital's size and capabilities. But what about geospatial questions? Such as:
    - How many hospitals do I have in my County?
    - How many residents of this area are within 25 miles of a hospital?
    - Disaster strikes: How many hospitals do I have within 100 miles?

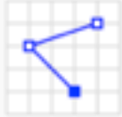
GIS Options

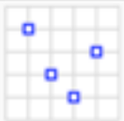
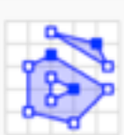
- We could parse out the City/State from our address, but this only allows us to count the number of hospitals per city/state.
- We could assign a latitude/longitude to each hospital. Issues with this:
  - We might have to write custom software to put on a map, off the shelf mapping options might not work.
  - Still need custom code for geospatial operations (Distance, Contains/Within/Intersects, etc…)
  - Could maybe have a solution for our Point data, but what about polygons and lines?

Enter PostGIS

- PostGIS adds support for geographic objects in Postgres: Points, LineStrings, Polygons and their “Multi” variations. Also supports Raster data (Digital Elevation Maps)
- Provides functions for determining geospatial interactions between objects (Within, Contains, Intersects) as well as measurements (Area, Length, Distance)
- Provides Spatial Indexes to vastly speed up geospatial operations
- Natively supported by off the shelf GIS tools: QGIS, Geoserver.

I stole this slide from wikipedia

Type	Examples	
Point		POINT (30 10)
LineString		LINESTRING (30 10, 10 30, 40 40)
Polygon		POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))
		POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))

Type	Examples	
MultiPoint		MULTIPOINT ((10 40), (40 30), (20 20), (30 10))
		MULTIPOINT (10 40, 40 30, 20 20, 30 10)
MultiLineString		MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))
MultiPolygon		MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))
		MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))

Example geom column

	name character varying(100)	admin character varying(200)	geom geometry(MultiPolygon,4326)
1	Aruba	Aruba	0106000020E6100000010000000103000000010000001A000000362D7CD3CD7F51C0
2	Badghis	Afghanistan	0106000020E6100000010000000103000000010000000301000005CAC4D5C99135040
3	Hirat	Afghanistan	0106000020E6100000010000000103000000010000006002000B006EA5F95AE4E40
4	Khost	Afghanistan	0106000020E6100000010000000103000000010000006900000003487EB5832845140
5	Pembroke	Bermuda	0106000020E6100000010000000103000000010000000A000000EE397CD3ED3150C0
6	Bamyan	Afghanistan	0106000020E6100000010000000103000000010000008101000009000CF2A9CEF5040
7	Kapisa	Afghanistan	0106000020E6100000010000000103000000010000004B000000AC52CFAA825B5140
8	Balkh	Afghanistan	0106000020E6100000010000000103000000010000006E010000502BF19295D05040
9	Ordino	Andorra	0106000020E6100000010000000103000000010000017000000800F04E9BD8EF93F
10	Canillo	Andorra	0106000020E6100000010000000103000000010000001F00000080B3AF5A822DFC3F
11	Faryab	Afghanistan	0106000020E6100000010000000103000000010000000BB01000080E584AC0E5D5040

Wall of text 1: Relationships and Measurements

- ST_Distance - Returns the distance between two objects (either 2D cartesian based on projection, or geodesic in meters depending on arguments)
- ST_Centroid - Returns geometric center of geometry
- ST_Disjoint - Returns TRUE if geoms don’t intersect
- ST_Intersects - Returns TRUE if geoms intersect
- ST_Contains - Returns TRUE if one geom is inside another
- ST_Length - Returns length of geom (either 2D cartesian based on projection, or geodesic in meters depending on arguments)

Example 1: Distance Between Cities

```
SELECT ST_Distance(
    (SELECT geom FROM geo.cities_ne WHERE name = 'Seattle'),
        (SELECT geom FROM geo.cities_ne WHERE name = 'New Orleans’),
    true) * 0.000621371 AS distance_miles
```

	distance_miles double precision
1	2099.5671408692 2

Wall of text 2: Geometry Accessors/Editors

- ST_Envelope - Returns the bounding box of the geometry
- ST_SRID - Returns the Spatial Reference Identifier for the geometry
- ST_Transform - Returns a new Geometry with its coordinates transformed to the desired SRID
- ST_Affine - applies affine transformations to the geometry
- ST_Rotate - Rotates a geometry about an origin.

Example 2: Coordinate Information and Transformation

- First we will get the SRID of one of our geometries:

```
SELECT ST_SRID((SELECT geom FROM geo.cities_ne WHERE name = 'Seattle'))
```

st_srid integer
4326

- Then we will transform it and observe the results

```
SELECT ST_AsText(
    ST_Transform(
        (SELECT geom FROM geo.cities_ne WHERE name = 'Seattle'),
        3857
    )
)
```

st_astext text
POINT(-13619041.444428 6035935.37594134)

Wall of text 3: Geometry Constructors and Outputs

- ST_GeomFromText - Creates a geom from WKT
- ST_GeomFromKML - Takes KML input and creates a geom
- ST_AsGeoJSON - Returns geom as GeoJSON
- ST_AsText - Returns Well Known Text Representation of geom

Getting data into PostGIS

- One of the easiest ways to get data into PostGIS is via Shapefiles and the utility shp2pgsql
- If you have raster data you want to put into PostGIS (Digital Elevation Maps is the common use case) you can use raster2pgsql

Example 3: Furthest Cities

Which two cities on earth are furthest from each other?

Example 3: Furthest Cities

```sql
SELECT
        c1.name || ', ' || c1.adm0name AS city1,
        c2.name || ', ' || c2.adm0name AS city2,
        ST_Distance(c1.geom, c2.geom, true) * 0.000621371 AS distance_miles

FROM geo.cities_ne c1
CROSS JOIN geo.cities_ne c2
WHERE c1.scalerank < 3 AND c2.scalerank < 3 --Limit the run time of
                                            --the query by focusing on
                                            --bigger cities
ORDER BY ST_Distance(c1.geom, c2.geom, true) DESC
LIMIT 10
```

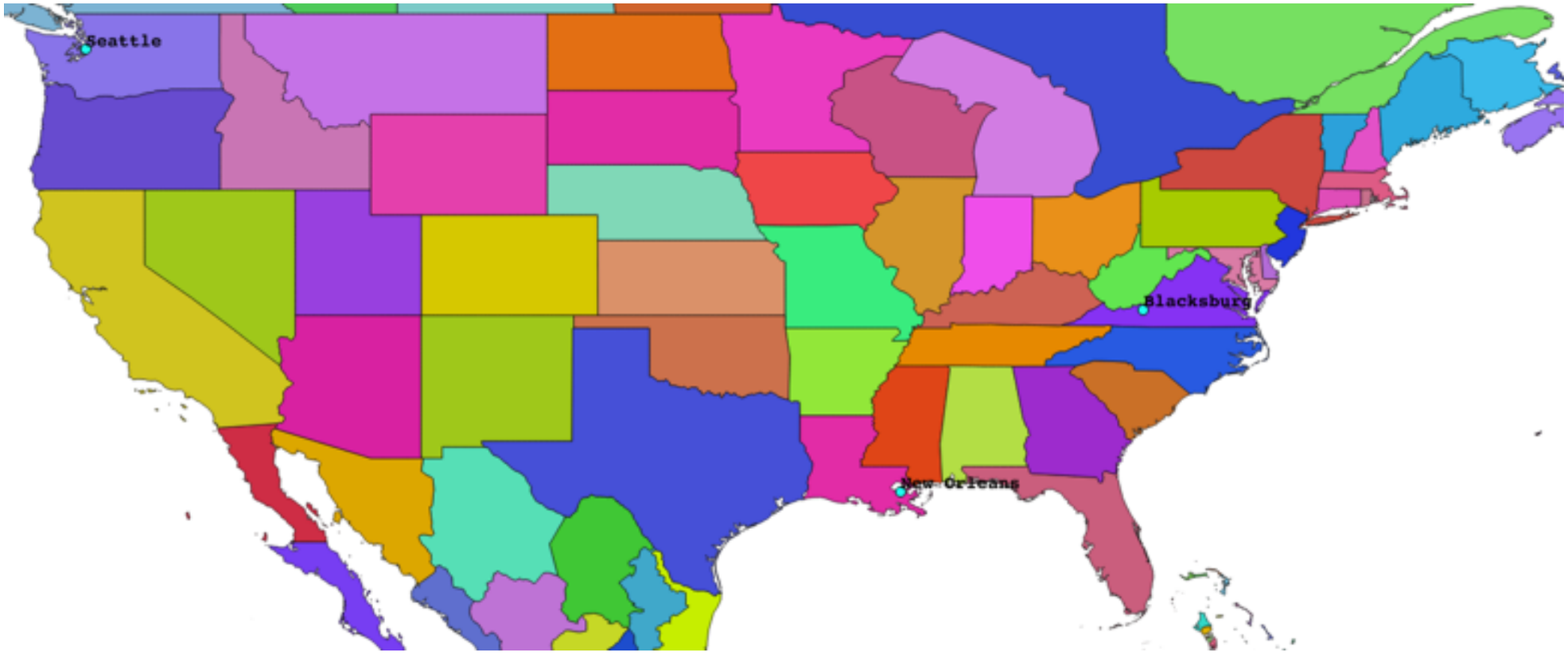
t pane

a Output | Explain | Messages | History

	city1 text	city2 text	distance_miles double precision
	Xian, China	Santiago, Chile	12369.0959635262
	Santiago, Chile	Xian, China	12369.0959635262
	Guayaquil, Ecuador	Kuala Lumpur, Malaysia	12314.341883938
	Kuala Lumpur, Malaysia	Guayaquil, Ecuador	12314.341883938

Example 4: Family Vacations

- My family lives all over the country, I'm in Seattle, my Sister is in Blacksburg Virginia, and my Parents are in New Orleans… Where is the fairest place for us to meet for Christmas?



Example 4: Solution

```
WITH fam_city AS(
        SELECT name, geom FROM geo.cities_ne c WHERE c.name = 'New Orleans'
        OR c.name = 'Blacksburg' OR c.name = 'Seattle'
)

SELECT c.name || ', ' || c.adm1name AS loc, ST_DISTANCE(c.geom,
        (SELECT ST_CENTROID(
                (SELECT ST_UNION(f.geom) FROM fam_city f))
        ), true) * 0.000621371 AS distance_miles

FROM geo.cities_ne c
ORDER BY ST_DISTANCE(c.geom, (SELECT ST_CENTROID(
                (SELECT ST_UNION(f.geom) FROM fam_city f)
        )), true)
```

t pane

a Output | Explain | Messages | History

loc text	distance_miles double precision
Hutchinson, Kansas	22.4276893479084
Salina, Kansas	38.5301663778016
Wichita, Kansas	40.4335715748189

LAB TIME

- Go to: http://maptimesea.github.io/2016/01/02/postgis-tutorial.html To start the lab!
- Slides and QGIS file for viewing lab data can be downloaded at https://github.com/parkercoleman/postgis_talk (Click Download Zip)
- I can be reached at ADINSX.sa@gmail.com for questions :)