Links

- http://www.manning.com/obe – We are co-authors of the upcoming **PostGIS in Action,** due out in hard-copy in January 2010. You can purchase now via the Manning Early Access Program (MEAP) and read as the tale unfolds. First chapter is a free download.

- http://www.bostongis.com – BostonGIS.com is focused on Open Source GIS tips and tricks.

- http://postgis.refractions.net – PostGIS core site – find everything PostGIS related here.

- http://www.postgresonline.com – Postgres OnLine Journal is focused on providing a resource for PostgreSQL users and newcomers by providing examples that demonstrate PostgreSQL's unique features, and how to use PostgreSQL effectively. Each edition available as free PDF.

- http://www.paragoncorporation.com – Homepage of our boutique database consulting company.

What is PostGIS?

- It is a **cost effective** alternative to Oracle Spatial/Locator, IBM DB2 Spatial, Informix Spatial Data Blade, and Microsoft SQL Server 2008. It shares many of the same characteristics as these other OGC SFSQL compliant products.

- PostGIS is a PostgreSQL module that adds OpenGIS Consortium (OGC) compliant geometry data types and functions to PostgreSQL.

- PostGIS is a GPL Open Source Project. Yes, it is fine for commerical use as long as you don't compile the PostGIS code into your binaries.

- It was developed by Refractions Research as a cost effective solution to managing spatial data; specifically for work they were doing with Canadian British Ministry. It has since then been enhanced and adopted by many companies.

How is it used?

- Database backend for web mapping and desktop applications
- Storage of spatial data
- Analysis of geographic data
  - Spread of disease around the world
  - Ecological changes
  - Correlating objects by space for property management, crime, emergency response, etc.
- Creation of smaller datasets for distribution
  - Extract data that fits within an arbitrary region of interest
  - Region tagging / geocoding data for distribution to non-spatial databases
  - Simplify data create lighter vector version for less work requiring less precision
  - Re-project data in different spatial reference system
- Fixing of geographic data
  - Fixing malform geometries you inherit from various sources

Who uses it?

- Government agencies
  - Property and building management
  - Environmental protection
  - Emergency response
  - Traffic control (air, land, sea)
  - Foreclosure prevention and forecasting

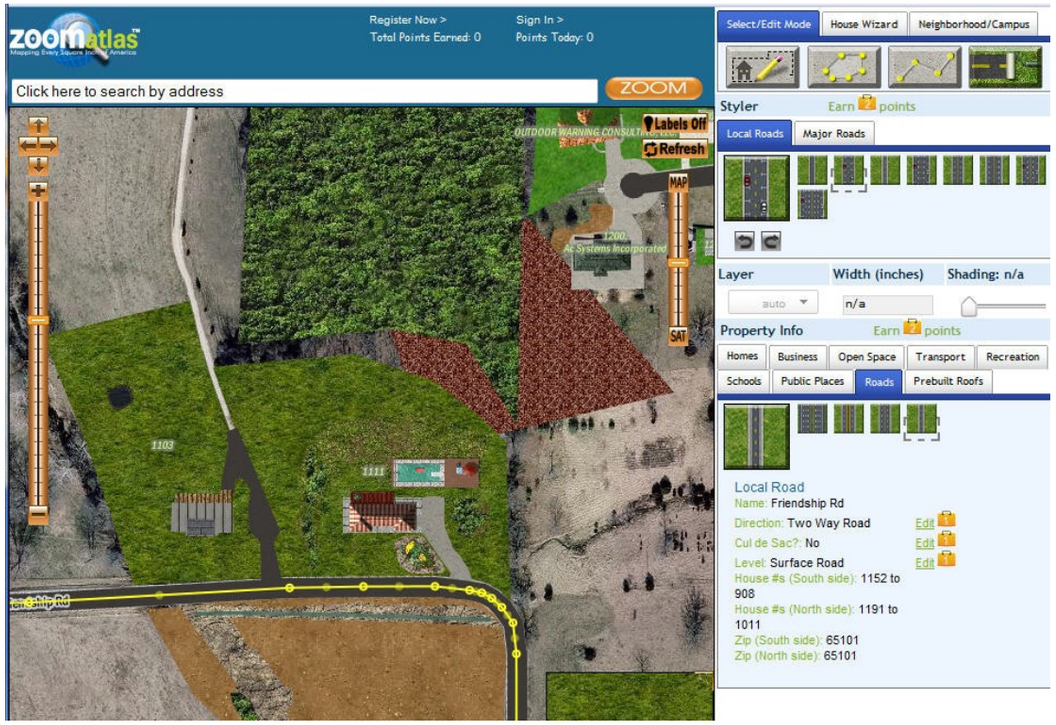
- Scientific research
  - Soil management
  - Ecology
  - Modeling and simulations

- Universities
  - GIS, urban development courses

- Private, political, labor sector
  - Fleet, salesforce management
  - Sales forecasting
  - Political districting
  - Grassroot organizing
  - Risk, hazard analysis for insurance

- Dot Coms
  - Developing communities around location-based awareness
  - Managing large amounts of spatial data to feed subscription services

Real-world Application
ZoomAtlas

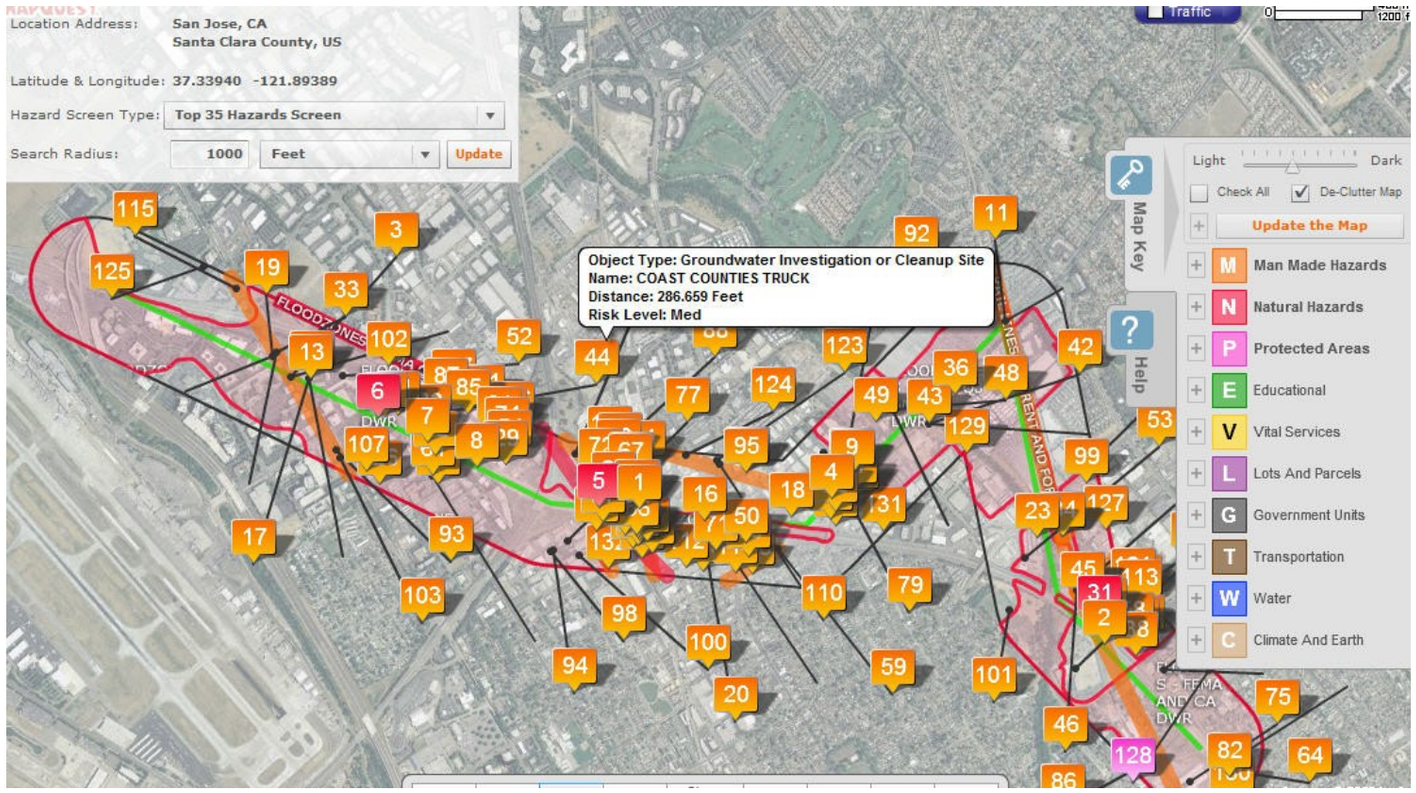


http://www.zoomatlas.com
Photorealistic Crowd Sourcing

- OpenLayers (WFS-T Client)
- Tomcat / GeoServer (WFS-T Server)
- Custom-built Tile cache

PostGIS backend to store street, houses, roads POIs of interest etc. Lot of data sourced from US Census Tiger and enhanced with other data and algorithms to automatically draw houses and other landscape features.

Real-world Application
Hazard Hunter



Mapping and Hazard scoring based on proximity to hazards and severity of hazards

- MapQuest Flex API,
- Degrapha
- Custom Flex
- PHP / ASP.Net

PostGIS used to store spatial data and returns spatial query results via custom web services

Tools to Load Data

Shp2pgsql

Command line data loader packaged with PostGIS
Imports standard ESRI Shapefiles and DBFs*

OGR2OGR http://fwtools.maptools.org/

Open source tool packaged in FWTools
Imports 20 different vector and flat file formats

* Good government sites for free data:
US: http://www.census.gov/geo/www/tiger
Canada: http://www12.statcan.ca/census-recensement/2006/geo/index-eng.cfm

Table Creation Statements

Heterogeneous Column Approach

```
CREATE TABLE places (
  place_id SERIAL PRIMARY KEY,
  place_name varchar(150),
  place_geom geometry
);

CREATE INDEX places_idx_place_geom ON places USING gist(place_geom);
```

Homogeneous Column Approach

```
CREATE TABLE assets.places (
  place_id SERIAL PRIMARY KEY,
  place_name varchar(150)
);

SELECT AddGeometryColumn('assets', 'places', 'place_geom', 26986, 'POINT', 2);

CREATE INDEX place_idx_the_geom ON assets.places USING gist(place_geom);
```

SRID 26986 is Massachusetts State Plane Coordinate System

Geometry Creation Statements

Making a point:

```
SELECT ST_SetSRID(ST_Point(-71.06737, 42.29586), 4326);
SELECT ST_GeomFromText('POINT(-71.06737 42.29586)', 4326);
SELECT
    ST_GeomFromWKB(E'\\001\\001\\000\\000\\000\\321\\256B\\312O\\304Q\\300\\347\\030\\220\\27
    5\\336%E@',4326);
SELECT CAST('0101000020E6100000D1AE42CA4FC451C0E71890BDDE254540' As geometry);
```

Other geometries:

A 2D linestring

```
SELECT ST_GeomFromText('LINESTRING(2 0,0 0,1 1,1 -1)');
```

A 3D multilinestring

```
SELECT ST_GeomFromEWKT('MULTILINESTRING((0 0 1,0 1 1,1 1 2),(-1 1 1,-1 -1 3))')
```

A multipoint

```
SELECT ST_GeomFromEWKT('MULTIPOINTM(-1 1 4,0 0 2,2 3 2)');
```

A square with holes

```
SELECT ST_GeomFromText('POLYGON((-0.25 -1.25,-0.25 1.25,2.5 1.25,2.5 -1.25,-0.25 -1.25),(2.25
0,1.25 1,1.25 -1,2.25 0),(1 -1,1 1,0 0,1 -1))'));
```

Creating Features

Create our schema

```
CREATE SCHEMA assets;
ALTER DATABASE pgcon2009 SET search_path=assets, public;
```

Create our tables

```
CREATE TABLE land(pid varchar(10) PRIMARY KEY, land_name varchar(150), land_type varchar(150));
SELECT AddGeometryColumn('land', 'the_geom', 26986, 'MULTIPOLYGON', 2);
CREATE INDEX assets_land_idx_the_geom ON land USING gist(the_geom);

CREATE TABLE building(gid SERIAL PRIMARY KEY, bldg_name varchar(150), bldg_type varchar(150));
SELECT AddGeometryColumn('building', 'the_geom', 26986, 'MULTIPOLYGON', 2);
CREATE INDEX assets_building_idx_the_geom ON building USING gist(the_geom);
```

```
CREATE TABLE residents(resid SERIAL PRIMARY KEY, pid varchar(10), income_level integer, num_adults
    integer, num_children_b12 integer, num_children_a12 integer);
ALTER TABLE residents
    ADD CONSTRAINT assets_residents_fk_land
    FOREIGN KEY (pid)
    REFERENCES land (pid) ON UPDATE CASCADE ON DELETE RESTRICT;
CREATE INDEX assets_residents_fki_land ON residents USING btree(pid);

CREATE TABLE hydrology(gid SERIAL PRIMARY KEY, hyd_name varchar(150), hyd_type varchar(150));
SELECT AddGeometryColumn('hydrology', 'the_geom', 26986, 'POLYGON', 2);
CREATE INDEX assets_hydrology_idx_the_geom ON hydrology USING gist(the_geom);

CREATE TABLE road(gid SERIAL PRIMARY KEY, road_name varchar(150), road_type varchar(150), nstart integer,
    nend integer);
SELECT AddGeometryColumn('road', 'the_geom', 26986, 'LINESTRING', 2);
CREATE INDEX assets_road_idx_the_geom ON road USING gist(the_geom);
```

Creating Roads

```
INSERT INTO road(road_name, road_type, the_geom, nstart, nend)
VALUES
('Main Rd', 'major', ST_GeomFromText('LINESTRING (247917 899350, 253267
    900217.7491572206, 255591 899424, 256791 897948,258359 897155, 259281
    897782, 259281 899738, 259392 900715, 253599 901564)', 26986), 1,
    1000),
('Curvy St', 'minor',
    ST_GeomFromText(ST_AsText(ST_CurveToLine('CIRCULARSTRING(257270
    897671,257224 897667,257178 897665,256695 897863,256489 898341)')),
    26986), 1, 100),
('Elephantine Rd', 'major',
    ST_SetSRID(ST_Translate(ST_Scale(ST_GeomFromText('LINESTRING(328 -8.5,
    323.5 -28.4, 328.1 -36.4, 320.7 -54.6, 331 -61, 340 -74, 340 -98, 361
    -103, 377 -99, 389.5 -98, 388.4 -89, 379 -88, 374 -68, 357 -46, 336
    -49,333 -36.4, 358 -31, 356 -5.6, 354 -7.9)'), 200,100), 190000,
    907000),26986), 1, 200000);
```

Creating Hydrology

```
INSERT INTO hydrology(hyd_name, hyd_type, the_geom)
VALUES
('Lake 1', 'lake', ST_GeomFromText('POLYGON ((254100 899740, 252280 898880,
    253080 898920, 254100 899740))', 26986)),
('Elephantine Youth', 'reservoir', ST_GeomFromText('POLYGON ((260298 900275,
    260969 897727, 264454 897995, 260298 900275))', 26986)),
('River 1', 'river', ST_Buffer(ST_GeomFromText('LINESTRING (254580 899820,
    253950.4022864219 899009.0145298447, 254480 898680, 254842.99706756207
    898301.1125329993, 254352.08176504684 898166.4503003974, 253960 898440,
    253020 898380, 253160 898120)', 26986), 20) ),
('Bigger River', 'river', ST_Buffer(ST_GeomFromText('LINESTRING (247752
    897838, 250869.32813777245 901306, 251275 900439, 253895 900827, 256053
    899424, 257898 898336, 258839 898521)', 26986), 15));
```

Adding Land

```
INSERT INTO land(pid, land_type, land_name, the_geom)
WITH RECURSIVE
p(pkey, atype, the_geom) AS
(
   VALUES (1, 'historical', ST_Multi(ST_Buffer(ST_Transform(ST_GeomFromText('POLYGON((-
   70.93052 42.31830,-70.93053 42.31840,-70.93053 42.31841,-70.93054 42.31838,-70.93052
   42.31830))',4326), 26986),100,1 ) ))
   UNION ALL
   SELECT pkey + 1, 'historical', ST_Multi(ST_Translate(the_geom, (ST_XMax(the_geom) -
   ST_XMin(the_geom)), (ST_YMax(the_geom) - ST_YMin(the_geom))))
   FROM p WHERE pkey < 20 )
   ,
p2(pkey,atype,the_geom) AS
(SELECT lpad(CAST(p.pkey + CAST(random()*100000 As integer) As text),9, '0') , (ARRAY['1
   family', 'condo', '2 family', '3 family','commercial', 'government', 'hospital',
   'police station', 'college', 'park', 'elementary school', 'highschool', 'vacant
   land'])[CAST(random()*12 As integer) + 1],
   ST_Multi(ST_Buffer(ST_Translate(p.the_geom,x*i, 2*pi()*sin(i/y) + (ST_YMax(the_geom) -
   ST_YMin(the_geom))) ,0,  mod(i,p.pkey) ))
FROM p CROSS JOIN (SELECT MAX(CAST(ST_XMax(the_geom) - ST_XMin(the_geom) As integer)) As
   x FROM p ) As x CROSS JOIN (SELECT Max(CAST(ST_YMax(the_geom) - ST_YMin(the_geom) As
   integer)) As y FROM p) As y
CROSS JOIN (SELECT CAST(2*n*sin(n*2/360.0) As integer) FROM generate_series(1,250) As n
   WHERE sin(n/360.0) <> 0) As i(i)
WHERE mod(i, p.pkey) between 1 and 4
)
SELECT pkey As pid, MIN(atype), MIN(atype) || pkey, MAX(the_geom)
FROM p2
WHERE ST_IsValid(the_geom)
GROUP BY pkey;
```

Remove unviable land

```
--Delete all land that gets in the way of our roads
-- , water and water ways
DELETE FROM land
   WHERE EXISTS (SELECT h.gid FROM hydrology As h WHERE ST_Intersects(h.the_geom,
   land.the_geom));

DELETE FROM land
      WHERE EXISTS (SELECT r.gid FROM road As r WHERE ST_Intersects(r.the_geom,
   land.the_geom));

--Delete all land that is not close enough to a road or water way
DELETE FROM land
      WHERE NOT EXISTS (SELECT h.gid FROM hydrology As h WHERE ST_DWithin(h.the_geom,
   land.the_geom, 25000));

DELETE FROM land
    WHERE NOT EXISTS (SELECT r.gid FROM road As r WHERE ST_DWithin(r.the_geom,
   land.the_geom, 3000));
```

Adding Building

```
INSERT INTO building(bldg_name, bldg_type, the_geom)
SELECT pid, land_type,
   ST_Multi(ST_Buffer(ST_ConvexHull(ST_Collect(ST_PointOnSurface(the_geom),
   ST_Centroid(ST_MinimumBoundingCircle(the_geom, 1 + mod(CAST(ST_Ymin(the_geom) As
   integer),4))))), (ST_XMax(the_geom) - ST_XMin(the_geom))/(5 + random()*10),1))
FROM  land
WHERE land_type NOT IN('vacant land', 'government');
```

Adding Residents

```
INSERT INTO residents(pid , income_level, num_adults, num_children_b12,
   num_children_a12 )
SELECT pid,
   CASE WHEN ST_Area(the_geom)/max_res > 5000 THEN 20000 + random()*100000
   WHEN ST_Area(the_geom)/max_res > 3000 THEN 10000 + random()*70000 ELSE random()*50000
   END,
   1 + CAST(random()*4 As integer), CAST(random()*12 As integer), CAST(random()*5 As
   integer)
FROM (SELECT pid, the_geom, land_type, CASE land_type WHEN '3 family' THEN 3 WHEN '2
   family' THEN 2 ELSE 1 END  As max_res
           FROM land) As l CROSS JOIN generate_series(1,3) As n
WHERE land_type LIKE '%family' or land_type LIKE '%residential%'
AND n <= max_res ;
```

OpenJump

Overview

- OpenJump - http://www.openjump.org
- Free Open Source (GNU GPL) Vector Desktop GIS tool. Very popular among hard-core PostGIS spatial database users
- Java 5+ based Cross Platform – can run on Linux/MacOSX/windows
- Key Features
  - Reads GML, SHP, DXF, MapInfo MIF, TIFF, JPG, MrSID, ECW, PostGIS and plug-ins available to support ESRI ArcSDE, Oracle, and MySQL
  - Can Write to GML, SHP, PostGIS, JML, JPG, PNG – plugins to write to autocad DXF
  - Great for rendering ad-hoc PostGIS spatial queries
  - OGC – support for WMS, WFS, GML 2, SLD
  - Basic Editing capability
  - Thematic Map Capability
  - Lots of tools for checking spatial quality of data and doing other spatial analysis and manipulations
  - Lots of custom plugins built by community users to do other stuff.
- We will be using it to do some ad-hoc queries here.
- Note: You will see us use ST_AsBinary function a lot. This is because the ad-hoc query window expects the geometry output to be in OGC binary format.

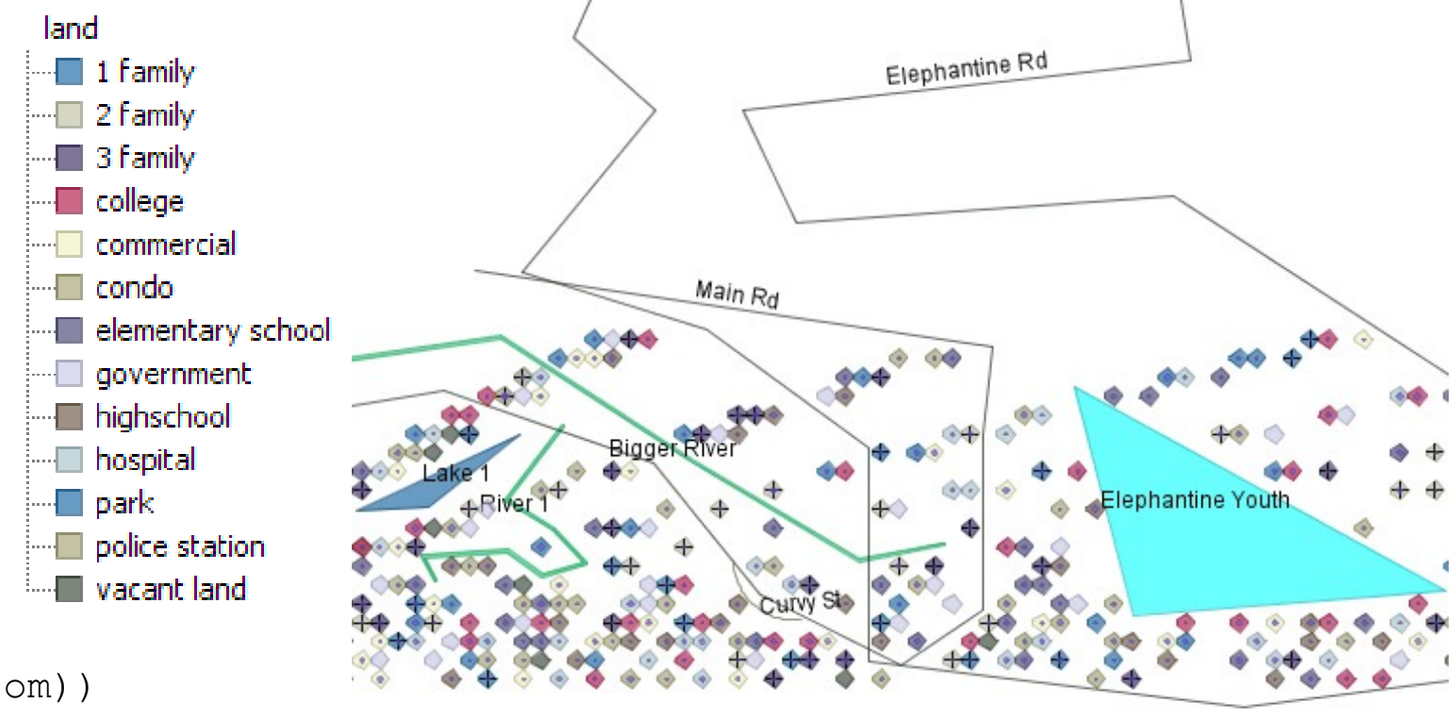
OpenJump
Our Town

- Hydrology

```
SELECT hyd_name, hyd_type,
    ST_AsBinary(the_geom)
FROM hydrology;
```

- Roads

```
SELECT road_name, road_type,
    ST_AsBinary(the_geom)
FROM road;
```



- Residents are crosses

```
SELECT r.income_level,
    ST_AsBinary(ST_Centroid(the_geom))
FROM residents As r
INNER JOIN land As l ON r.pid = l.pid;
```

- Land are colorful polygons to distinguish land type

```
SELECT ST_AsBinary(the_geom), land_type
FROM land;
```

- Buildings are polygons inside land

```
SELECT ST_AsBinary(the_geom)
FROM building;
```

Our Town
What a mess: Legal

We have land boundary legal issues. This query tells us that.

```
--Return the area of land we have, union dissolves overlaps
SELECT SUM(ST_Area(the_geom))/1000 As totalwithoverlap_km,
ST_Area(ST_Union(the_geom))/1000 As no_overlap_km
FROM land;
```

```
 totalwithoverlap_km |  no_overlap_km
---------------------+-----------------
    18172.1253334656 | 13135.024279953
```

We even have buildings smashing into each other.

```
SELECT SUM(ST_Area(the_geom))/1000 As totalwithoverlap_km,
ST_Area(ST_Union(the_geom))/1000 As no_overlap_km
FROM building;
```

```
  totalwithoverlap_km |  no_overlap_km
----------------------+------------------
    840.745522918701 | 682.593583892822
```

Our Town
Can relationships help?

Which parcels of land intersect others and with what (use new array_agg in 8.4)

```
--Return the pids of parcels that intersect other parcels
SELECT p.pid, COUNT(o.pid) As totinter, array_agg(o.pid) As inter_parcels
FROM land As p
    INNER JOIN land As O ON (p.pid <> o.pid AND ST_Intersects(p.the_geom,
   o.the_geom))
GROUP BY p.pid
ORDER BY p.pid;
```

```
  pid      | totinter   |              inter_parcels
-----------+------------+-----------------------------------------
 000000113 |          2 | {000088531,000023521}
 000000421 |          4 | {000028675,000067999,000078713,000040546}
 000000680 |          4 | {000099217,000006903,000092497,000032961}
 000001038 |          1 | {000093591}
:
```

What kind of intersection?

```
SELECT COUNT(o.pid) As totinter, COUNT(CASE WHEN ST_Overlaps(o.the_geom,p.the_geom) THEN
   1 ELSE NULL END) As o_overlaps_p,
   COUNT(CASE WHEN ST_Equals(o.the_geom,p.the_geom) THEN 1 ELSE NULL END) As o_eq_p
FROM land As p
    INNER JOIN land As O ON (p.pid <> o.pid AND ST_Intersects(p.the_geom, o.the_geom));
```

```
  totinter | o_overlaps_p | o_eq_p
-----------+--------------+--------
       758 |            0 |    758
```

Our Town Cleanup 1

Last query proved all our problems are caused by dupe land

```
--Residents reassigned to non-dupe land
UPDATE residents
    SET pid = a.newpid
    FROM (SELECT p.pid, MIN(o.pid) As newpid
                FROM land As p INNER JOIN land As O ON
                        (p.pid = o.pid OR
   ST_Equals(p.the_geom, o.the_geom))
                GROUP BY p.pid
                HAVING p.pid <> MIN(o.pid)) As a
    WHERE residents.pid = a.pid;
```

-- 135 residents reassigned

Our Town Cleanup 2

Get rid of duplicated parcels and merge info into one

--Create new field to house new types

```
ALTER TABLE assets.land ADD COLUMN land_type_other varchar(150)[];
```

--copy all additional land_types to first parcel

```
UPDATE land
   SET land_type_other = a.dupe_types
   FROM (SELECT p.pid, MIN(o.pid) As newpid, array_agg(DISTINCT o.land_type) as
   dupe_types
            FROM land As p INNER JOIN land As O ON
                  (ST_Equals(p.the_geom, o.the_geom))
            GROUP BY p.pid
            HAVING COUNT(p.pid) > 1 AND p.pid = MIN(o.pid)) As a
   WHERE land.pid = a.pid;

--delete remaining dupe parcels
DELETE FROM land
   WHERE pid IN
    (SELECT p.pid
            FROM land As p INNER JOIN land As O ON
                  (ST_Equals(p.the_geom, o.the_geom))
            GROUP BY p.pid
            HAVING COUNT(p.pid) > 1 AND p.pid <> MIN(o.pid)) ;
```

Our Town Cleanup 3

```
--Our  building issue is more complicated as buildings are not dupes
-- create new building table
CREATE TABLE newbuilding(gid SERIAL PRIMARY KEY, bldg_name text, bldg_type text);
SELECT AddGeometryColumn('newbuilding', 'the_geom', 26986, 'MULTIPOLYGON', 2);

--Copy records and union where appropriate
INSERT INTO newbuilding(gid, bldg_name, bldg_type, the_geom)
SELECT b.gid, array_to_string(array_agg(o.bldg_name), '|'), array_to_string(array_agg(o.bldg_type), '|'),
ST_Multi(ST_Union(o.the_geom))
FROM building As b INNER JOIN building As o
ON (ST_Intersects(b.the_geom, o.the_geom))
GROUP BY b.gid
HAVING b.gid = MIN(o.gid);

--Compare to make sure we didn't loose realestate
SELECT SUM(ST_Area(the_geom)), ST_Area(ST_Union(the_geom))
FROM newbuilding;

SELECT ST_Area(ST_Union(the_geom))
FROM building;

--drop old
SELECT DropGeometryTable('building');

--reinstate new
ALTER TABLE newbuilding RENAME TO building;
UPDATE geometry_columns SET f_table_name = 'building'
  WHERE f_table_name = 'newbuilding';

CREATE INDEX assets_building_idx_the_geom ON building USING gist(the_geom);
vacuum analyze building;
vacuum analyze land;
vacuum analyze residents;
vacuum analyze hydrology;
vacuum analyze road;
```

Our Town
Distance checks

```
--What percentage of kids under the age of 12
--are further than half mile of an elementary school?
SELECT SUM(num_children_b12)*100.00/(SELECT
   SUM(num_children_b12) FROM residents)
FROM  residents As r
INNER JOIN land As l ON r.pid = l.pid
LEFT JOIN (SELECT pid,  the_geom FROM land
                WHERE land_type = 'elementary school'
                     OR  'elementary school' =
   ANY(land_type_other) ) As eschools
         ON ST_DWithin(l.the_geom, eschools.the_geom,
   1609/2)
WHERE  eschools.pid IS NULL;
```

--For this simulation 17%

Our Town
Spatial space checks

--Of the land that have buildings how many have greater than 1000 sq meters left.

```
SELECT COUNT(lb.pid)
FROM  (SELECT l.pid, l.the_geom As land_geom,
  ST_Union(b.the_geom) As bldg_geom FROM
     land As l INNER JOIN building As b
 ON ST_Intersects(l.the_geom, b.the_geom)
        GROUP BY l.pid, l.the_geom) AS lb
WHERE ST_Difference(lb.land_geom, bldg_geom) >
  1000;
```

Our Town
Linear Interpolation and NN

```
--What kinds of buildings can we find within
-- 100 meters of Bigger River
-- note we only consider buildings that are within 500 meters of their closest road
-- and what is the closest interpolated address on that road, land use, and distance from  road address interp
--if we want long lat we can add
-- ST_AsText(ST_SnapToGrid(ST_Transform(b_loc,4326),0.0001)) As as_text_b_loc
SELECT h.nstart +
      CAST( (h.nend - h.nstart)
                  * ST_Line_Locate_Point(h.street_line, ST_Centroid(h.b_loc)) As integer)
                  As street_num , h.road_name, h.land_use,
                  CAST(ST_Distance(h.street_line, h.b_loc) As numeric(10,2)) As dist_road
FROM
(SELECT array_to_string(bl.land_type || bl.land_type_other,',') As land_use , bl.street_line,
    ST_Centroid(bl.the_geom) As b_loc, bl.nstart ,
      bl.nend, bl.road_name
FROM (SELECT DISTINCT ON(b.gid) l.*, b.gid, e.the_geom As street_line, e.nstart, e.nend, e.road_name
    FROM
      building As b INNER JOIN land As l ON ST_Intersects(b.the_geom, l.the_geom)
   INNER JOIN road As e ON ST_DWithin(e.the_geom, l.the_geom,500)
   ORDER BY b.gid, ST_Distance(e.the_geom,l.the_geom) ) As bl
  INNER JOIN hydrology As w ON ST_DWithin(w.the_geom, bl.the_geom, 100)
WHERE w.hyd_name = 'Bigger River'
) As h
ORDER BY street_num, road_name;
```

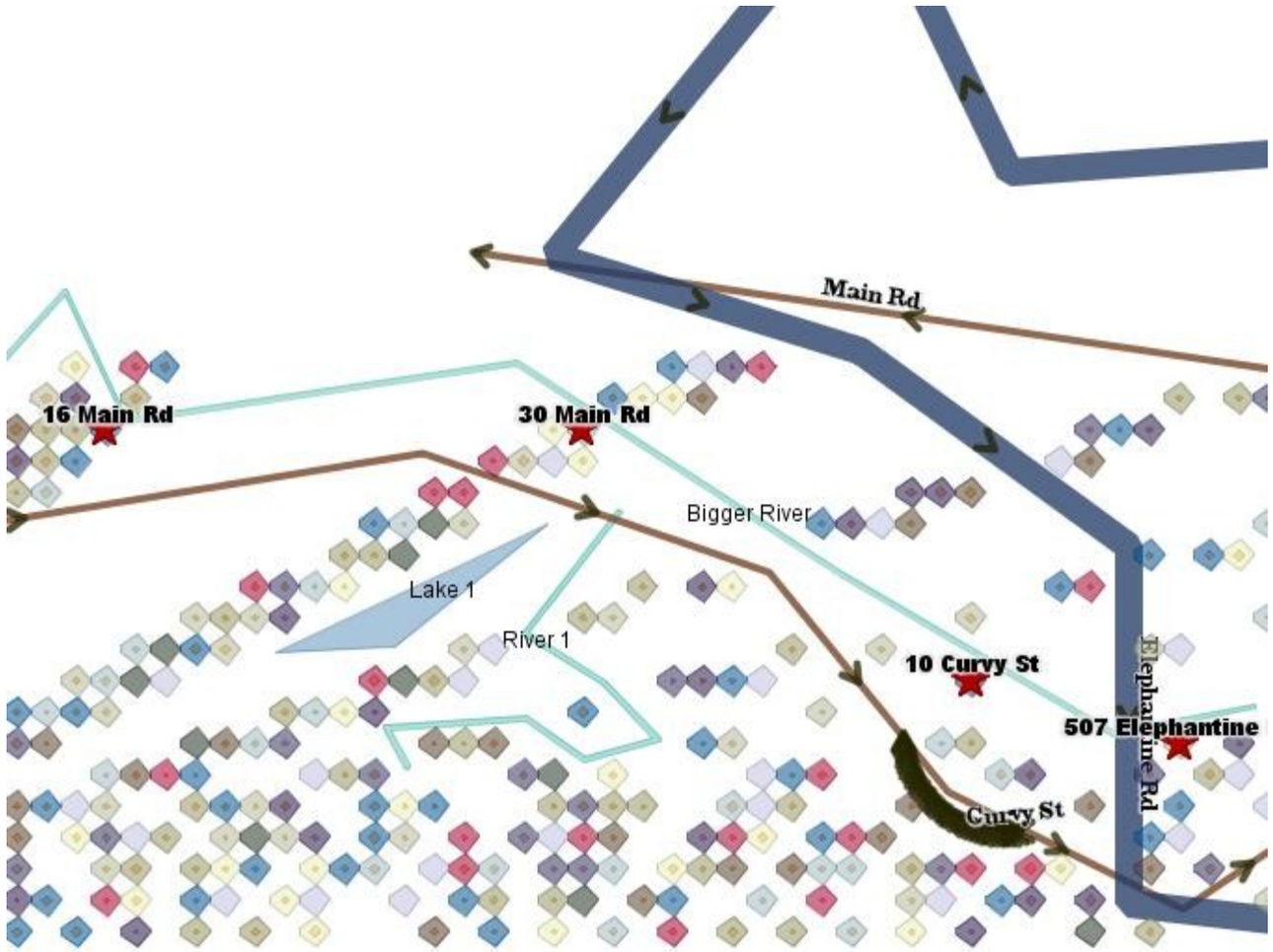
street_num	road_name	land_use	dist_road
10	Curvy St	elementary school	570.29
16	Main Rd	park,condo,park,police station	500.89
30	Main Rd	hospital	497.45
507	Elephantine Rd	2 family	337.74

Our Town
Plot on Open Jump

```
SELECT ST_AsBinary(h.b_loc) As binloc,
    CAST(h.nstart +
       CAST( (h.nend - h.nstart)
                    *
    ST_Line_Locate_Point(h.street_line,
    ST_Centroid(h.b_loc)) As integer) As
    text) || ' ' || h.road_name As address,

    CAST(ST_Distance(h.street_line, h.b_loc)
    As numeric(10,2)) As dist_road
FROM
(SELECT array_to_string(bl.land_type ||
    bl.land_type_other,',') As land_use ,
    bl.street_line,
    ST_Centroid(bl.the_geom) As b_loc,
    bl.nstart ,
       bl.nend, bl.road_name
FROM (SELECT DISTINCT ON(b.gid) l.*, b.gid,
    e.the_geom As street_line, e.nstart,
    e.nend, e.road_name FROM
      building As b INNER JOIN land As l ON
    ST_Intersects(b.the_geom, l.the_geom)
   INNER JOIN road As e ON
    ST_DWithin(e.the_geom, l.the_geom,500)
   ORDER BY b.gid,
    ST_Distance(e.the_geom,l.the_geom) ) As
    bl
  INNER JOIN hydrology As w ON
    ST_DWithin(w.the_geom, bl.the_geom, 100)
WHERE w.hyd_name = 'Bigger River'

) As h;
```



Our Town Windowing

```
-- For each resident on Elephantine Rd
-- Number sequential in order by street number,resid
-- provide their income level and the income level of their neighbor in the  door before and the door after
SELECT row_number() OVER (ORDER BY sr.street_num, sr.resid) As row_num, sr.resid, sr.street_num,
  sr.income_level,
  lag(sr.income_level, 1) OVER( ORDER BY sr.street_num, sr.resid) As prevd_inc,
  lead(sr.income_level, 1) OVER( ORDER BY sr.street_num, sr.resid) As nextd_inc
FROM (SELECT DISTINCT ON(r.resid) r.resid, r.income_level, array_to_string(l.land_type ||
    l.land_type_other,',') As land_use,
h.nstart + CAST( (h.nend - h.nstart)
                  * ST_Line_Locate_Point(h.the_geom, ST_Centroid(l.the_geom)) As integer) As
    street_num, h.road_name
FROM
      land As l INNER JOIN residents As r ON l.pid = r.pid
   INNER JOIN road As h ON ST_DWithin(h.the_geom, l.the_geom,500)
   ORDER BY r.resid, ST_Distance(h.the_geom,l.the_geom) ) As sr
WHERE sr.road_name = 'Elephantine Rd'
ORDER BY street_num, road_name;
```

row_num	resid	street_num	income_level	prevd_inc	nextd_inc
1	97	315	104872		26972
2	259	315	26972	104872	92683
3	421	315	92683	26972	59742
4	113	387	59742	92683	82924
5	275	387	82924	59742	21183
6	437	387	21183	82924	83568
7	115	394	83568	21183	59635
8	277	394	59635	83568	92089
9	439	394	92089	59635	98719

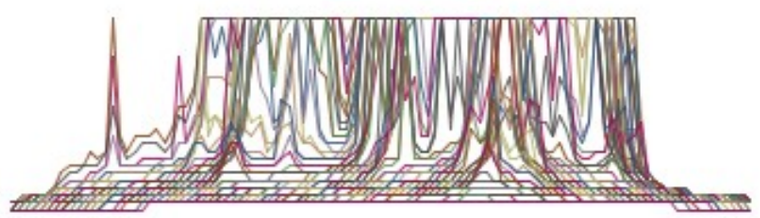
:
:

Questions

Spatial SQL Art

A child's drawing with Recursive

```
INSERT INTO mypois(poi_name, poi_geom)
WITH RECURSIVE
x(i)
AS (
    VALUES(0)
UNION ALL
    SELECT i + 1 FROM x WHERE i < 101
),
Z(Ix, Iy, Cx, Cy, X, Y, I)
AS (
    SELECT Ix, Iy, X::float, Y::float, X::float, Y::float, 0
    FROM
        (SELECT -2.2 + 0.031 * i, i FROM x) AS xgen(x,ix)
    CROSS JOIN
        (SELECT -1.5 + 0.031 * i, i FROM x) AS ygen(y,iy)
    UNION ALL
    SELECT Ix, Iy, Cx, Cy, X * X - Y * Y + Cx AS X, Y * X * 2
     + Cy, I + 1
    FROM Z
    WHERE X * X + Y * Y < 16.0
    AND I < 27
),
Zt (Ix, Iy, I) AS (
    SELECT Ix, Iy, MAX(I) AS I
    FROM Z
    GROUP BY Iy, Ix
    ORDER BY Iy, Ix
)
SELECT 'mandlebrot ' || CAST(Iy As text),
     ST_MakeLine(ST_MakePoint(Ix,I)) As poi_geom
FROM Zt
GROUP BY Iy
ORDER BY Iy;
```



OpenJump Query:

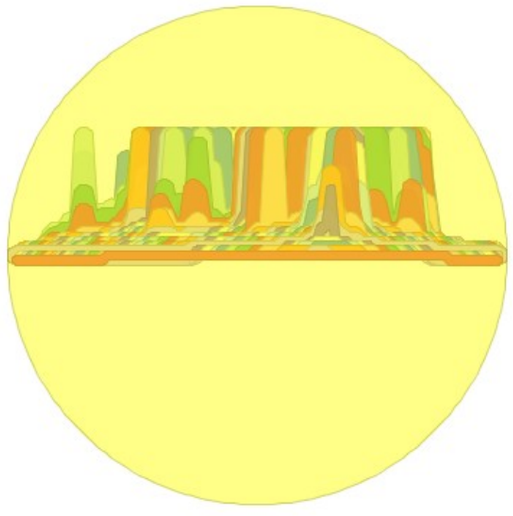
```
SELECT poi_name,
ST_AsBinary(poi_geom)
FROM mypois
WHERE poi_name LIKE 'mandlebrot%'
```

Spatial SQL Art

A child's drawing Buffered and Framed

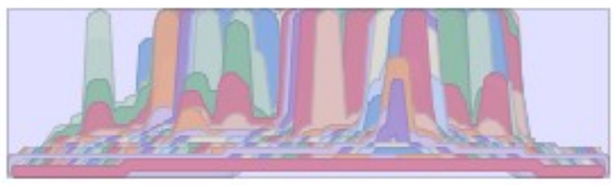
(1) OpenJump Query:

```
SELECT poi_name, ST_AsBinary(ST_Buffer(poi_geom,
ST_YMin(poi_geom)))
FROM mypois
WHERE poi_name LIKE 'mandlebrot%';
```



(2) OpenJump Query 2: Okay now we add another layer to frame in its minimum bounding circle

```
SELECT
ST_AsBinary(ST_MinimumBoundingCircle(ST_Collect(
ST_Buffer(poi_geom, ST_YMin(poi_geom))
    )
  )
)
FROM mypois
WHERE poi_name LIKE 'mandlebrot%';
```



(3) Convex Hull:

```
SELECT
ST_AsBinary(ST_ConvexHull(ST_Collect(ST_Buffer(poi_
geom, ST_YMin(poi_geom)))))
FROM mypois
WHERE poi_name LIKE 'mandlebrot%';
```

(4) Just the Extent:

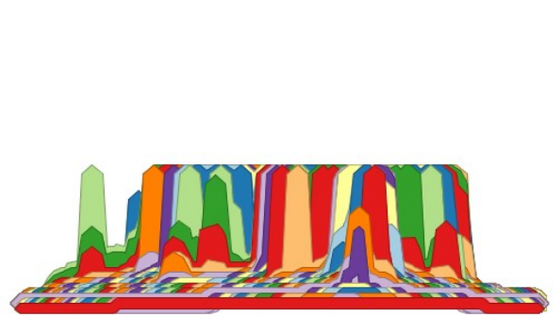
```
SELECT
ST_AsBinary(ST_Extent(ST_Buffer(po
i_geom, ST_YMin(poi_geom))))
FROM mypois
WHERE poi_name LIKE 'mandlebrot%';
```

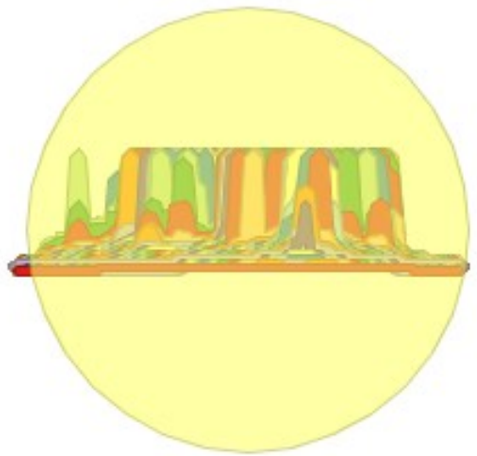
Spatial SQL Art

More Buffers

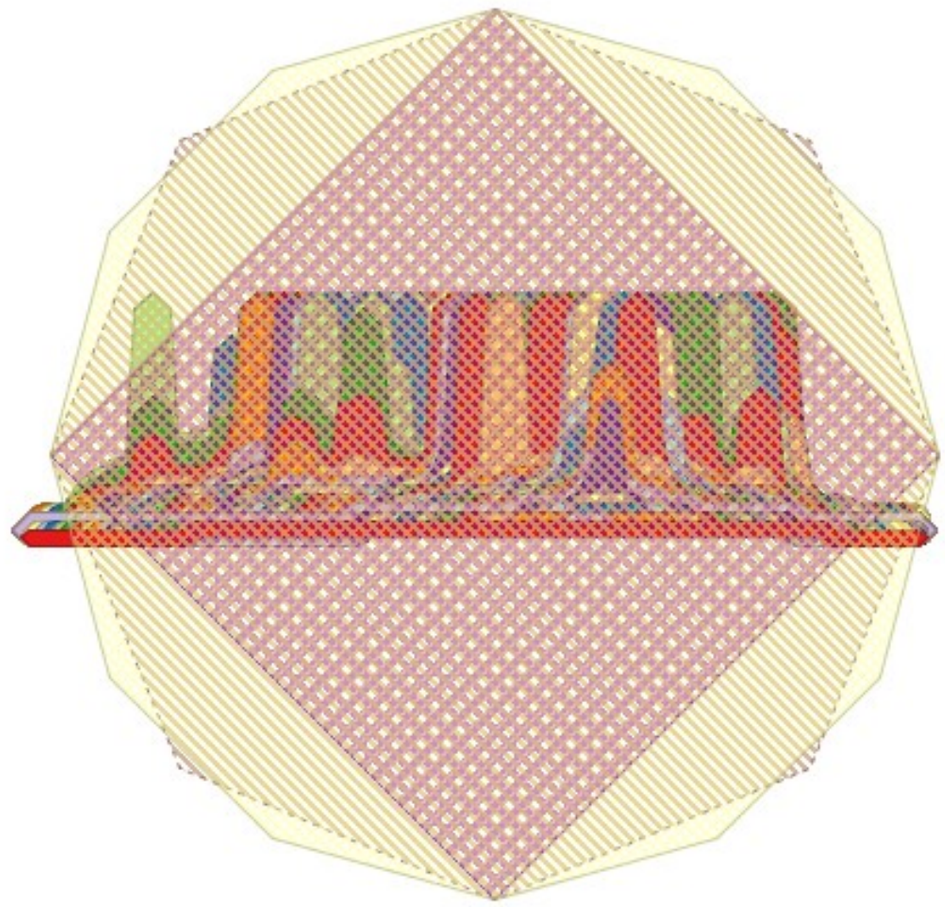
(1) OpenJump Query:

```
SELECT poi_name,
ST_AsBinary(ST_Buffer(poi_geom,
ST_YMin(poi_geom),1))
FROM mypois
WHERE poi_name LIKE 'mandlebrot%';
```





```
(2) SELECT ST_AsBinary(
ST_Buffer(ST_Centroid(
    ST_Collect(
          ST_Buffer(poi_geom,
ST_YMin(poi_geom),1))),
Max(greatest(ST_XMax(poi_geom)/2,
ST_YMax(poi_geom)/2))))
FROM mypois;
```



```
(3) SELECT n, ST_AsBinary(
ST_Buffer(ST_Centroid(
    ST_Collect(
          ST_Buffer(poi_geom,
ST_YMin(poi_geom),1))),
Max(greatest(ST_XMax(poi_geom)/2,
ST_YMax(poi_geom)/2)), n))
FROM mypois CROSS JOIN
generate_series(1,3) As n
GROUP BY n;
```