PostgreSQL & PostGIS Showcase - Centralized spatial data manipulation, storage and retrieval

As a new user of GIS tools and data, it became evident that the huge volumes of raster and vector based data files you acquire, manipulate and modify can quickly get out of control. Developing a server based system which can use to access GIS data is desirable.

PostgreSQL is a full featured, mature, open sourced RDBMS which implements ANSI SQL standards. PostGIS is a separate product which adds support for geographic objects to PostgreSQL. This combination offers optimized spatial queries following specifications developed and standardized by the Open Geospatial Consortium (OGC). Analogous alternatives to PostGIS include ESRI SDE & Oracle Spatial Extensions.

This presentation will give an overview of PostGIS, using sample GIS datasets and open sourced GIS tools to access and display GIS information stored in a PostgreSQL remote database. This platform offers a low cost solution to help implement GIS with Registry data.
Actual Search results for all .SHP files on one user’s machine.

Each entry shows a distinct copy of a shape file (directory path omitted)
PostgreSQL

- Mature ANSI compliant, SQL RDBMS
  - (ie MS SQLServer, Oracle, MySQL, …)
- Open Source
- Well integrated administration GUI called PGAdmin III
PostGIS

- Integrated ‘Add On’ for PostgreSQL
- Supports spatial functions and processing.
- Integrates with PostgreSQL GiST & ‘Sparse indexes’
- Widely supported by ‘Open’ GIS projects
- Import and Export Shape files
- (raster support in future versions)
PostgreSQL & PostGIS

- Runs on Win32, Xnix, OsX, Solaris platforms.
- Linux install my preference. (Ubuntu long term support (LTS) Server good solution.
- My environment uses precompiled, Ubuntu APT packages (ie EASY to install !)

 Сан Diego, June 2009
Setting up a Spatial Database

1) Create a database. You must be the `postgres` superuser to do this:

From the shell:

```bash
$ createdb mytestdb
```

**TIP:** if you get permission errors, try:

```bash
$ sudo su postgres
$ createdb mytestdb
```

**REMEMBER:** type 'exit' when you're done with these postgres commands to return to your normal user.

2) Set up the postgis libs:

```bash
$ createlang plpgsql mytestdb
```

2.a) If you installed from packages:

```bash
$ psql -d mytestdb -f /usr/share/postgresql-8.2-postgis/lwpostgis.sql
```

if there were no errors (if the last line of output is COMMIT), then

```bash
$ psql -d mytestdb -f /usr/share/postgresql-8.1-postgis/spatial_ref_sys.sql
```

*San Diego, June 2009*
PostgreSQL gui admin tool – showing a Spatially Enabled PG database
DDL to define a new PostGIS table

-- Table: "CalifBlockGroups"

-- DROP TABLE "CalifBlockGroups";

CREATE TABLE "CalifBlockGroups"
(
    gid integer NOT NULL,
    "STATEFP00" character varying(2),
    "COUNTYFP00" character varying(3),
    "TRACTCE00" character varying(6),
    "BLKGRPCE00" character varying(1),
    "BKGPIDFP00" character varying(12),
    "NAMESAD00" character varying(13),
    "MTFCC00" character varying(5),
    "FUNCSTAT00" character varying(1),
    the_geom geometry,
    CONSTRAINT "CalifBlockGroups_pkey" PRIMARY KEY (gid),
    CONSTRAINT enforce_dims_the_geom CHECK (ndims(the_geom) = 2),
    CONSTRAINT enforce_srid_the_geom CHECK (srid(the_geom) = (-1))
)
WITH (OIDS=FALSE);

ALTER TABLE "CalifBlockGroups" OWNER TO postgres;

CREATE INDEX "CalifBlockGroups_ndx"
ON "CalifBlockGroups"
USING gist
(the_geom);

San Diego, June 2009
DDL to define a new PostGIS table

-- Table: "CalifBlockGroups"

-- DROP TABLE "CalifBlockGroups";

CREATE TABLE "CalifBlockGroups"
(
    gid integer NOT NULL,
    "STATEFP00" character varying(2),
    "COUNTYFP00" character varying(3),
    "TRACTCE00" character varying(6),
    "BLKGRPCE00" character varying(1),
    "BKGPIDFP00" character varying(12),
    "NAMELSAD00" character varying(13),
    "MTFCC00" character varying(5),
    "FUNCSTAT00" character varying(1),
    the_geom geometry,
    CONSTRAINT "CalifBlockGroups_pkey" PRIMARY KEY (gid),
    CONSTRAINT enforce_dims_the_geom CHECK (ndims(the_geom) = 2),
    CONSTRAINT enforce_srid_the_geom CHECK (srid(the_geom) = (-1))
) WITH (OIDS=FALSE);
ALTER TABLE "CalifBlockGroups" OWNER TO postgres;

CREATE INDEX "CalifBlockGroups_ndx"
ON "CalifBlockGroups"
USING gist
(the_geom);
Load a SHP file

Intact forests data files – from Green Peace

San Diego, June 2009
You could use Command Line tools:

Load data into PostgreSQL from ESRI shape file:

```
shp2pgsql -s 4326 world_ifl_block30 public.worldforesttest > forestsload.sql
psql -h myserver -d mydb -U myuser -f forestsload.sql
```
...Or – use a plugin called SPIT, in an open source tool called QGIS
San Diego, June 2009
Viewing the uploaded SHP file, using QGIS tool
OGC Influence

- Open Geospatial Consortium
- Non profit, voluntary consensus standards body
- Publish and help implement ‘specs’. Such as
  - • **WMS** - Web Map Service
  - • **WFS** - Web Feature Service
  - • **WCS** - Web Coverage Service
  - • **CAT** - Web Catalog Service
  - • **SFS** - Simple Features for SQL
  - • **GML** - Geography Markup Language
OGC Influence

- Open Geospatial Consortium
- Non profit, voluntary consensus standards body
- Publish and help implement ‘specs’. Such as
  - • WMS - Web Map Service
  - • WFS - Web Feature Service
  - • WCS - Web Coverage Service
  - • CAT - Web Catalog Service
  - • SFS - Simple Features for SQL
  - • GML - Geography Markup Language
SQL – the familiar query language - has specific, GIS extensions for the GIS community, which are laid out by OpenGIS PostGres/PostGIS – it is called Simple Features for SQL (SFSQL)

What is SFSQL?

One of the greatest things about Spatial Relational Databases is that they bring GIS to a new level by allowing you to apply the expressive SQL declarative language to the spatial domain. With spatial relational databases, you can easily answer questions such as what is the average household income of a neighborhood block, what political district does X reside in. This new animal that marries SQL with GIS is called Simple Features for SQL (SFSQL). In essence SFSQL introduces a new set of functions and aggregate functions to the SQL Language.
COMMON USE SFSQL EXAMPLES

--Create a spatial index on the new geometry column
ALTER TABLE testtable ALTER COLUMN the_geom SET NOT NULL;
CREATE INDEX idx_testtable_the_geom ON testtable USING gist(the_geom);
ALTER TABLE testtable CLUSTER ON idx_testtable_the_geom;

--Create a geometry column named the_geom in a
--table called testtable located in schema public
--to hold point geometries of dimension 2 in WGS84 longlat
SELECT AddGeometryColumn('public', 'testtable', 'the_geom', 4326, 'POINT', 2);

--Insert a record into the new table
INSERT INTO testtable(description, the_geom)
VALUES('center of boston',
       ST_GeomFromText('POINT(-71.0891380310059, 42.3123226165771)', 4326));

--Insert a point record into the new table - faster than st_geomfromtext for points
INSERT INTO testtable(description, the_geom)
VALUES('center of boston',
       ST_SetSRID(ST_MakePoint(-71.0891380310059, 42.3123226165771), 4326));

--Break up multipolygons into individual polygons
SELECT neigh_name,
       ST_GeometryN(the_geom, generate_series(1, numgeometries(the_geom))) AS polygeom
FROM neighborhoods;

--Take individual polygons and create one multipolygon for each neighborhood
--Note if you have a mixed collection of geometries, will return a geometry collection
SELECT neigh_name, ST_Collect(polygeom) as the_geom
FROM neighborhoods
GROUP BY neigh_name;

San Diego, June 2009
Common Spatial Type (ST) Functions

Accessors

- ST_Dimension
- ST_Dump
- ST_EndPoint
- ST_Envelope
- ST_ExteriorRing
- ST_GeometryN
- ST_GeometryType
- ST_InteriorRingN
- ST_IsClosed
- ST_IsEmpty
- ST_IsRing
- ST_IsSimple
- ST_IsValid
- ST_mem_size
- ST_M
- ST_NumGeometries
- ST_NumInteriorRings
- ST_NumPoints
- ST_npoints
- ST_PointN
- ST_SetSRID
- ST_StartPoint
- ST_Summary
- ST_X
- ST_XMin, ST_XMax
- ST_Y
- ST_YMin, ST_YMax
- ST_Z
- ZMin, ZMax

Geometry Processors

- ST_Boundary*
- ST_Buffer*
- ST_BuildArea*
- ST_Centroid+
- ST_ConvexHull*
- ST_Difference*
- ST_Expand
- ST_ForceRHR
- ST_Union*
- ST_Intersection*
- ST_PointOnSurface*
- ST_Reverse
- ST_RotateX
- ST_RotateY
- ST_RotateZ
- ST_Scale
- ST_Simplify
- ST_SymDifference*
- ST_Transform
- ST_Translate
- ST_TransScale

Measurement

- ST_Area
- ST_Azimuth
- ST_Distance
- ST_distance_sphere
- ST_distance_spheroid
- ST_length_spheroid
- ST_length
- length3d_spheroid
- ST_max_distance
- ST_Perimeter

Outputs

- ST_AsBinary
- ST_AsText
- ST_AsEWKB
- ST_AsEWKT
- ST_AsHEXEWKB
- ST_AsGML
- ST_AsKML
- ST_AsSVG

San Diego, June 2009
Smallest area of any Calif tract

```sql
SELECT "TRACTCE00", ST_Area(the_geom)
FROM "CalifTracts"
order by ST_Area(the_geom) limit 1;
```

San Diego, June 2009
Smallest area of any Calif BG

```
SELECT "BKGPIDFP00", ST_Area(the_geom)
FROM "CalifBlockGroups"
order by ST_Area(the_geom) limit 1;
```
Tracts WITHIN a specific distance from a given point – SB Fire

SELECT gid, "STATEFP00", "COUNTYFP00", "TRACTCE00" FROM "CalifTracts" where st_dwithin(the_geom, 'point(-119.7247 34.4480)', .005);
QGIS delimited text import plugin

Description
Select a delimited text file containing a header row and one or more rows of x and y coordinates that you would like to use as a point layer and this plugin will do the job for you!

Use the layer name box to specify the legend name for the new layer. Use the delimiter box to specify what delimiter is used in your file (e.g. space, comma, tab or a regular expression in Perl style). After choosing a delimiter, press the parse button and select the columns containing the x and y values for the layer.

Delimited text file: C:/GIS/PluginImport.txt
Layer name: PluginImport
Delimiter: | (Plain characters)
X field: longitude
Y field: latitude
Sample text:
latitude,longitude,depth,bl
34.448|-119.7247|0|Fire
Recent Santa Barbara Wildfire

QGIS tool showing Census tracts, plus imported ‘coordinate’ file with fire location
Links and Resources

PostgreSQL:
http://www.postgresql.org/

PostGIS:
http://postgis.refractions.net/

Good PostGIS tutorial:
http://www.bostongis.com/?content_name=postgis_tut01

Documentation:
http://postgis.refractions.net/documentation/manual-1.3/
http://www.bostongis.com/postgis_intersection_intersects.snippet

Great QGIS references:
http://blog.qgis.org/blog/3
Links and Resources

Geospatial SQL system – feature matrix/comparison


Open Geospatial Consortium:

http://www.opengeospatial.org/

Future for RASTER storage on PostgreSQL:


http://trac.osgeo.org/postgis/wiki/WKTRaster
Good introductory reference:
http://tinyurl.com/r54f4y
HOT off the press – 118 pg draft
http://www.manning.com/obe
For their patience and help during all the nagging, dumb, redundant questions along the way…. Many Thanks to:

Myles C, Dan G – and all the NAACCR GIS committee members

San Diego, June 2009