Application of PostGIS database management on spatial data and enabling them in Mapserver environment

By: Arash G.Mojaveri
230029192
GEOG 413- Advanced GIS
Instructors: Dr. Roger Wheate – Scott Emmons –Ping Bai

December, 2004

Abstracts

The following report is the first part of a joint project course of GEOG413 (advanced GIS). The first step is to enable spatial data in order to be presented with in Mapserver. (http://mapserver.gis.umn.edu/). To do so, we need two software; Postgresql (pronounce post-Grez-ql) and PostGIS.

Introduction

In order to enable spatial data queries with in Mapserver (online access to maps and its attributes and capability of asking questions as well as manipulating them) without using GIS application such as ArcInfo. Postgres is open source database software which has both UNIX and Micro Soft Windows ports available free to download from ftp://ftp.PostgreSQL.org/pub. It has all the features, performance, reliability of other database systems plus some additional advantages such as user defined types, inheritance and multi-version concurrency control. The study area is in Tanzania, Africa and we need to create a database of birds spices then transfer that database in to Mapserver so it can be joint to some functionality from PostGIS and finally show the spatial queries on the database and show them by Mapserver.

Data Source

The data were provided by Dave Lemon. He is an independent researcher working with other people from Tanzania trying to monitor the birds life over there. A pdf file containing the related information was given to us by him.

Data Manipulation

PostGIS by Refractions Research Inc (Victoria, BC, Canada –a data integration and custom software development) is an extension to the PostgreSQL object-relational database system which allows GIS objects to be stored in to database. It includes support for GiST-based R-Tree spatial indexes(R-Tree is one of relational database structures to manage files in order to make access to data less expensive. An R-tree index is used for indexing spatial data. A hash index can't handle range searches. A B-tree index only handles range searches in a single dimension. R-trees can handle multi-dimensional data), and functions for analysis and processing of GIS objects.
For more information on setting up a PostGIS database refer to:
http://datashare.gis.unbc.ca/wiki/doku.php?id=datashare:postgis

The package needs the followings in order to transfer GIS files into a database.
a) A complete configured and built PostgreSQL multi-version concurrency control source code tree.
b) A "C" compiler ("`gcc`" GNU has a better performance)
c) A make file ("`make`" or "`gmake`")
GEOS geometry and Proj4 reprojection libraries might be needed as well to convert different projections to what we need.

In order to compile PostGIS (as an extension to PostgreSQL), we need to have a full copy of the PostgreSQL. So we need to compile and install both PostgreSQL first and PostGIS next. PostGIS requires the PL/pgSQL procedural language extension. Before loading the `postgis.sql` file, one must first enable PL/pgSQL by "`createlang`" command (adding a new programming language to a PostgreSQL database).
http://pgsqld.active-venture.com/app-createlang.html

PL/pgSQL needs to be enabled: `# createlang plpgsql databaseName`

Then we load the PostGIS object and function definitions into the database by loading the `postgis.sql` definitions file.
`# psql -d [databasename] -f postgis.sql`
In order to have a set of coordinate system definition identifiers, we also load the "`spatial_ref_sys.sql`" definitions file and populate the `SPATIAL_REF_SYS` table (this will be taken care of by PostGIS).
`# psql -d [databasename] -f spatial_ref_sys.sql`
At this point PostGIS server extension is ready. PostGIS object could be of any type of point, line, polygon, multipoint, multiline, multipolygon, and geometry collections.
PostGIS uses Simple Features Specification (SFS) of OpenGIS for SQL. It defines standard GIS object types, the functions required to manipulate them, and a set of meta-data tables.
All operations on spatial columns like deleting or adding a column are done by OpenGIS since they have special procedures defined for doing such. So we can make sure about the data to be consistent.
OpenGIS has two meta-data tables: SPATIAL_REF_SYS table and GEOMETRY_COLUMNS table. The SPATIAL_REF_SYS table keeps the IDs and descriptions of coordinate systems used in the spatial database. SRID in this table is an integer number which refers to the coordinate system used for that specific geometry. The GEOMETRY_COLUMNS table is a meta-data table meaning that it is a table which holds the information regarding other tables. Here we can see some of the tables:

```
ghasemza@ninkasi:~> psql -d ghasemza
Password:
Welcome to psql 7.4.2, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

ghasemza=> \d
                 List of relations
 Schema |             Name             |   Type   |  Owner
--------+------------------------------+----------+----------
 public | boundary_poly                | table    | ghasemza
 public | clip                         | table    | ghasemza
 public | clip_gid_seq                 | sequence | ghasemza
 public | clipbirds                    | table    | ghasemza
 public | fcover                       | table    | ghasemza
 public | fcover2                      | table    | ghasemza
 public | fcover_point                 | table    | ghasemza
 public | fcover_poly                  | table    | ghasemza
 public | fcoverclip                   | table    | ghasemza
 public | forestc                      | table    | ghasemza
 public | forestclip                   | table    | ghasemza
 public | geometry_columns             | table    | ghasemza
 public | pgis_unbc_clip               | table    | ghasemza
 public | spatial_ref_sys              | table    | ghasemza
 public | tanzbirds                    | table    | ghasemza
 public | unbc_bound_pgis              | table    | ghasemza
 public | unbc_bound_pgis_contains     | table    | ghasemza
 public | unbc_bound_pgis_int           | table    | ghasemza
 public | unbc_bound_pgis_intersectiona | table    | ghasemza
 public | unbc_bound_pgis_overlaps      | table    | ghasemza
(20 rows)
```

I tried to have them under my own Schema but for some reasons it was moved to public Schema (even though I have had the “create Schema” SQL query in “ghasemza” Schema) it was yet created under public Schema database. For a list of attributes and detail descriptions of them please refer to manual under documents directory in http://postgis.refractions.net/docs/.

```
ghasemza=> select * from Geometry_columns;
 f_table_catalog | f_table_schema |  f_table_name   | f_geometry_column | coord_dimension |  srid  |     type     | attrelid | varattnum | stats | unbc_bound_pgis_intersectiona
-----------------+----------------+-----------------+-------------------+-----------------+--------+--------------+----------+-----------+-------+------------------------------
                 | public         | tanzbirds       | geom              |               2 |   4133 | MULTIPOINT   | 12442598 |        12 |       |
                 | public         | birds           | geom              |               2 |   5133 | MULTIPOINT   |          |           |       |
                 | public         | xydave_point    | geom              |               2 |     -1 | MULTIPOINT   |          |           |       |
                 | public         | clip            | the_geom          |               2 |   5133 | MULTIPOLYGON | 12595630 |         3 |       |
                 | public         | unbcforestclipt | geom              |               2 |     -1 | MULTIPOLYGON |          |           |       |
                 | public         | fcover2         | geom              |               2 |  32610 | MULTIPOLYGON | 12597123 |       277 |       |
                 | public         | fcover          | geom              |               2 |  32610 | MULTIPOLYGON | 12598234 |        93 |       |
                 | public         | forestclip      | geom              |               2 |  32610 | MULTIPOLYGON | 12598660 |         5 |       |
                 | public         | fcover_point    | geom              |               2 | 104324 | MULTIPOINT   | 12599378 |        91 |       |
                 | public         | fcover_poly     | geom              |               2 | 104325 | MULTIPOLYGON | 12599680 |        93 |       |
                 | public         | boundary_poly   | geom              |               2 | 104326 | MULTIPOLYGON | 12600073 |         7 |       |
(11 rows)
```

```
ghasemza=> select * from spatial_ref_sys;
srid  | auth_name | auth_srid |
                                                                                       srtext

                                                  |                                                                                proj4text

--------+-----------+-----------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------+-----------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------
   2000 | ESRI      |      2000 | PROJCS["Anguilla_1957_British_West_Indies_Grid",GEOGCS["GCS_Anguilla_1957",DATUM["D_Anguilla_1957",SPHEROID["Clarke_1880_RGS",6378249.145,293.465]],PRIMEM["Greenw
ich",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Transverse_Mercator"],PARAMETER["False_Easting",400000],PARAMETER["False_Northing",0],PARAMETER["Central_Meridian",-62],PARAMETER["Scale_Fa
ctor",0.9995000000000001],PARAMETER["Latitude_Of_Origin",0],UNIT["Meter",1]]
                                                  | +proj=tmerc +lat_0=0 +lon_0=-62 +k=0.999500 +x_0=400000 +y_0=0 +ellps=clrk80 +units=m
         ……
```

One row of the table of “tanzbirds” is :

```
ghasemza=> select * from tanzbirds;
species                         | cardno | square | month_ | bno | status | longitude | latitude  |  yr   | ab | obs_nr |      geom
--------------------------------+--------+--------+--------+-----+--------+-----------+-----------+-------+----+--------+--------------------------------------------
Silvery-cheeked Hornbill |     71    | 3308D |      1     | 509 | P      |   33.5625 |   3.4375 | 1985 |  0 |      4     | SRID=4133; MULTIPOINT(33.5625 3.4375)
```

And one row of the table of “fcover_poly” which is the forest cover polygon of the UNBC area in Prince george before applying any PostGIS function on it to get the clipped area:
(this table is just for the testing purpose and has no information about my project)

```
ghasemza=> select * from fcover_poly;
area     | perimeter | fcover_ | fcover_id |   mapstand   | stand |   fcode    | fc_tag | mapsheet_id | forest_cover_pol | polygon_area | non_productive_f | non_productive_c | esa_category1_cd
 | esa_category2_cd | esa_wildlife_cd | card_source_cd | input_date | projected_date | coast_interior_c | pri_util_lvl_cd | sec_util_lvl_cd | inoperability_in | nts_map_number | layer_cnt | histor
y_cnt | resultant_cnt | for_cover_rank_c | for_cover_layer_ | non_forest_descr | data_srce_class_ | blank1 | data_srce_orign_ | species_cd_1 | species_pct_1 | species_cd_2 | species_pct_2 | specie
s_cd_3 | species_pct_3 | species_cd_4 | species_pct_4 | species_cd_5 | species_pct_5 | species_cd_6 | species_pct_6 | vol_per_ha_spp_1 | vol_per_ha_spp_2 | vol_per_ha_spp_3 | vol_per_ha_spp_4 | vo
l_per_ha_spp_5 | vol_per_ha_spp_6 | mean_diameter_pr | mean_diameter_se | site_index_estim | type_identity_re | stand_age | age_class_cd | age_range_minimu | age_range_maximu | update_reference |
inventory_type_g | stand_height | height_class_cd | volume_adjustmen | hist_class_site_ | hist_class_speci | air_crown_closur | stems_per_hectar | well_spaced_stem | stocking_class_c | stocking_cl
ass_s | history_referenc | reference_year | projected_age | projected_height | projected_type_i | projected_age_cl | projected_stocki | site_index | site_class_5m | stand_establishm | growth_type_
grou | crown_closure_cl | dbh_limit_cd | culmination_mai_ | stand_ba_pri_lvl | stand_ba_sec_lvl | projected_ba_pri | projected_ba_sec | secondary_elemen | e00_centroid_y | e00_centroid_x | SRID

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------
       67167 |   1851.93 |       2 |         1 | 093G096  341 |   341 | FI00000000 | 960341 | 093G096     |             341 |         17.3 | U                |               54 |
 |                 |                 | FA               |     980706 |           101 | I                |                4 |                8 |                  |                |         1 |
    0 |             2 |               1 |                1 |                  |             0 |         |             0 |               |             0 |               |             0 |
       |             0 |              |             0 |              |             0 |              |             0 |               0 |               0 |               0 |               0 |
             0 |               0 |               0 |               0 |                 |               6 |         0 |            0 |               0 |               0 |                 |
                 |            0 |              0 |               0 |                 |                 |               0 |               0 |               0 |                 |
      |                 |            94 |            0 |               0 |                6 |               0 |                 |          0 |            0 |               0 |
     |               0 |              |               0 |               0 |               0 |               0 |               0 |                 |    5.97392e+06 |         509106 | SRID=10
4325;MULTIPOLYGON(((508852.387000004 5973938.39,508849.479000002 5973908.944,508851.589999995 5973869.992,508849.097999999 5973855.7,508851.166999997 5973839.743,508870.600000006 5973826.021,50889
4.843000003 5973827.341,508907.539000001 5973816.186,508916.783000004 5973782.224,508916.721000001 5973761.853,508929.252000002 5973746.767,508947.722000003 5973741.858,508975.025999996 5973746.89
8,508990.450999995 5973752.61,509007.275000001 5973769.44,509030.146999997 5973768.737,509059.639999999 5973762.311,509085.283000003 5973771.776,509102.111000001 5973789.381,509099.862000006 59738
07.555,509086.280000003 5973811.119,509060.369000001 5973811.913,509043.643000001 5973813.21199999,509035.436000006 5973833.902,509013.036 5973844.011,508997.124999995 5973849.064,508982.741999996
 5973854.824,508969.454000003 5973870.619,508965.294999997 5973883.881,508965.743999996 5973897.79099999,508977.391999996 5973926.125,508995.923999996 5973934.087,509019.493 5973928.119,509035.425
000001 5973922.349,509063.696999999 5973921.619,509079.761999997 5973921.96999999,509115.604000005 5973917.216,509144.485000004 5973911.05499999,509158.999000003 5973911.416,509187.383000005 59739
17.517,509205.786000005 5973920.834,509270.994000002 5973916.235,509304.049999999 5973916.865,509330.334999997 5973912.527,509346.648000003 5973912.36499999,509369.240999997 5973915.969,509398.060
999995 5973924.768,509400.913999995 5973953.005,509408.191999995 5973972.702,509431.452999995 5973982.134,509454.009000002 5973995.39299999,509455.855673876 5974010.09537694,509250.747614747 59740
20.4102529,509231.722999995 5974021.367,509191.078000001 5974016.305,509147.425000004 5974012.86,509113.784000006 5974011.467,509068.705999999 5974006.622,509071.984999997 5973988.027,509061.72599
9997 5973972.974,509048.636999998 5973968.71499999,509030.362000005 5973971.496,508987.927999994 5973972.87299999,508967.144000002 5973963.494,508945.816999998 5973961.826,508925.171999998 5973957
.815,508907.618000002 5973955.257,508893.913999999 5973954.129,508867.295999999 5973952.623,508852.387000004 5973938.39)))
```

In order to put GIS objects in to the database we need a table with column of type "geometry". After creating the table with the required names and types for each column (attributes), we need to bring the data into our database table. There are two ways to get data into a PostGIS/PostgreSQL database.1- using formatted SQL statements, 2- using the Shape file loader. I tried to use "loader". The loader is called "shp2pgsql" and converts ESRI shape file into SQL suitable for loading in PostGIS/PostgreSQL.

```
# shp2pgsql [shapefile] [table] [database] > *.sql
# psql -d [database] -f *.sql
# psql -d [database] -f spatial_ref_sys.sql
```

The function "AddGeometryColumn" needs to be invoked with the proper arguments: [AddGeometryColumn(<db_name>, <table_name>, <column_name>,
<srid>, <type>, <dimension>)].
“srid” is the ID of the spatial reference system used for the coordinate geometry in this

table. It is a foreign key reference to the SPATIAL_REF_SYS.

At this point I had a problem to invoke that function. The function existed but was not able to be invoked. So I have tried another approach to transfer the GIS shape file information in to a database table as the base of my job. One of the Canadian software that is known to be powerful to transform different spatial formats is FME. http://www.safe.com/
Using FME workbench, I created a new workspace with the source format of ESRI shape

file and destination format of PostGIS Database. In FME, PostGIS dataset is treated as
database containing a collection of relational tables and their corresponding geometries. "When reading from the PostGIS/PostgreSQL database, each table is considered a feature type in FME and each row of a table at least one feature in FME. In the case of geometry collections, they may become more than one FME feature. The basic reading process involves opening a connection to the database, querying metadata, and querying data. The data is read using a text cursor and rows are fetched to the client machine in batches of 10000 by default. There is one cursor per input table. If NULL geometries are read, they are treated as non-geometry features and the attributes are preserved"(from FME help).

The database hierarchy structure is as next page diagram. The main server is Ninkasi and PostgreSQL and PostGIS are the database servers under it. The name of the Schema that I was creating the Tanzania birds’ database was “ghasemza” (my user name).

Spatial Analysis Methods

To avoid any incorrect result by applying irrelevant psql queries which might end up to wrong answers in mapserver ultimately, we decided to figure out which functions are equivalent to ESRI's functions such as clip, intersect and so on. This process was done in two steps based on the nature of the two features in spatial data bases.

Test on point features
Test on polygon features

For the point features, the result of a clip of "tanzbirds.shp" from ArcGIS was compared to a clip of the same data in tabular form from PostGIS. The original data for this file were gathered by a third party source from Africa and transferred to a shape file. The file

is information about different bird species existing in Tanzania, Africa.
In order to make a clip with ArcGIS, I used ArcMap. After creating an empty shape file with the type of polygon as my overlay layer (GEOG300-lab6), I clipped it with the base layer which was" tanzbirds.shp". The result was a clip by ArcGIS.
To make an empty polygon in PostGIS I was supposed to use the same shape file from ArcGIS and then using the following command transferring it to PostGIS:
(should be in the same directory as the database)

```
"shp2pgsql -s 5133 –c birdsclippoly.shp public.clip > upload.sql "
```

-s = SRID;
-c= create a new table and upload the data into that table;
public= name of the schema in data base
But since the "loader" function was not working properly in the software, I used FME to change the clip from GIS shape file to PostGIS data file.
Then I used a function from PostGIS package called "within" which is one of Geometry Relational functions in PostGIS library.
Function `within (geometry, geometry)` which returns true (1) if this Geometry is spatially within another Geometry. The "psql" command is:

```
"create table clipbirds(like tanzbirde)";
```

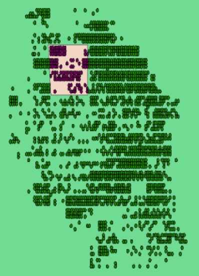
// this will make an exact copy of the table "tanzbirds" but empty
then we put the data of our clip in to the table:

```
insert into clipbirds(select * from tanzbirds where within(tanzbirds.geom, clip.the_geom) ;
```

and finally

```
"pgsql2shp –f newshapeBirds –p password ghasemza clipbirds"
```

// this will transfer the tabular form of the clip to a shape file. Once again the loader did not work properly, so I used FME to change the clipped in PostGIS to ESRI shape file.
At this point we can compare the two shape files using "ArcMap" to find out if we have the same results from ArcGIS and PostGIS.
Comparing the two point shape files did not show any difference or lack of data both in attributes and number of records.
Then I needed to do the same about the polygons.
The second test was on polygon features. I chose forest cover as my base layer and UNBC boundary as overlay. This needs to be compared with a shape file that will be transferred from a PostGIS database as a table. The way to do it is same as what was described for point features but a bit difference in tow ways
1- Specifying the source file in FME to change from PostGIS to shape file.
2- The function to use from PostGIS library in order to do the clip of the polygons.
One of the mistakes that I have made during the transformation was that I did not realize that the two files that I was using had different Projection sources. So at the end when I tried to apply functions from PostGis library, such as "intersection", it returned empty files. So the first thing is to make sure all the data have the same projections.
So I repeated the clipping with new data sets in ArcMap. The base layer was "fcover" ArcInfo coverage file and the overlay layer was UNBC boundary. Notice that fcover was



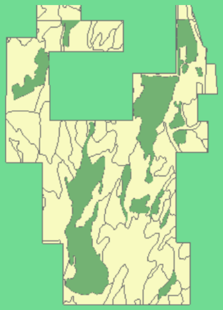
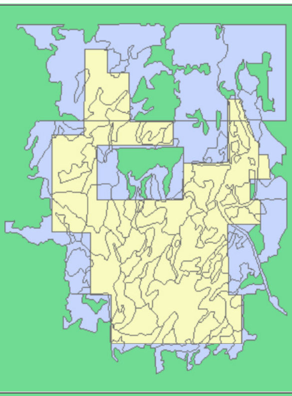
a) The result of a clipping "tanzbirds" and "polygon" using ArcInfo
b) The result of a clipping "tanzbirds" and "polygon" using "within" function of PostGIS.

It shows that this is exact same as clipping in ArcMap.

a coverage layer. Then FME was applied to change the two to a PostGIS format. The source format is ESRI coverage for fcover and shape for UNBC boundary. We need to change the setting of the destination as follows: hostname: Ninkasi, Port: 5432, username: ghasemza, password: ****** each time to let the software know where to put the translated data.
Then run the FME to return the translation. Repeat the process for both fcover and boundary. The next step is to apply one of the functions in PostGIS to get a clip from these two tables. The function "within" was applied same as above for point features.

```
#   create table fcoverclip(like fcover_poly);
#   insert into fcover (select * from fcover_poly where within
                       (fcover_poly.geom, boundary_poly.geom));
```

Then I applied FME to translate the clipped polygon from PostGIS to shape file. Finally using the ArcMap showed that the two polygon shape files (original clip from ArcInfo and translated one from PostGIS) have same attributes but do not have the same number of entries. The reason was that we did not apply the right PostGIS function. The "within" function does not include those polygons from overlay that have some parts of out of boundary. Then I used function "intersects". In this case the number of polygons was more than what I had from ArcInfo. The reason was that the "intersects" does not have the capability of cutting the polygons that are both in overlay and base layer. There were some other functions such as "crosses" or "touches" or "intersection" but neither of them gave me the right answer.
Now we can apply the data base toward the Mapserver and using the library functions of PostGIS write the queries on spatial data.

	
a) The result (green)from a clipping UNBC boundary and "fcover" using "within" function of PostGIS.	b) The result (blue) from clipping UNBC boundary and "fcover"

as I mentioned, it is not the right answer. using "intersects" function. The result was not cut from the edges.

Analysis Results, conclusions and future developments / works

The comparisons between the point features from ArcInfo and PostGIS database shows that we can apply the functions (such as within) from Postgresql instead of using "clip" from ArcInfo while all the attributes and quantity and other spatial information are totally preserved. The only problem which was remained was that due to the time constrain, I was advised to submit it for now and continue the rest of it on polygon features and the relative functions to do the queries on them for my independent study during the next semester.

References

The sources which had been used in this report:
http://postgis.refractions.net/
http://www.postgresql.org/
http://www.safe.com/
http://datashare.gis.unbc.ca/wiki/doku.php?id=datashare:postgis

The whole project was under supervision of Scott Emmons, the GIS database manager and lab instructor. Thanks to him.

TOP