UPGRADING TO POSTGIS 2.0 IN THE BRAZILIAN FEDERAL POLICE FORENSICS GIS

D. Miranda, D. Russo and R. A. L. Alves

Environmental Forensics Department, National Institute of Criminalistics
Brazilian Federal Police, Edificio INC, room A 121, 70610-200, Brasilia – DF, Brazil
(russo.dr, miranda.dam, rafael.rala)@dpf.gov.br

Commission IV/VIII

KEY WORDS: Updating Core Spatial Databases, PostGIS 2.0, PostgreSQL, GIS

ABSTRACT:

Inteligeo, the Geospatial database of the Brazilian Federal Police Forensics Department, contains more than 800 layers and almost 7 million features. It was initially stored in a PostgreSQL database using the PostGIS 1.5 spatial extension. Most of the data in the system was obtained due to police intelligence field work and lacks standardization and metadata. Sometimes the data itself contains inaccuracies and topological errors. In some cases these errors are easily correctable but PostGIS 1.5 does not have tools to make these basic corrections automatically. Another common day-to-day task is calculating areas, distances and lengths. Since the standard spatial reference system is in geographical coordinates it was necessary to find a suitable projection for the area to obtain the measurements in metric units. PostGIS 1.5 does not have a tool to automatically choose an adequate projection. Looking into the newest PostGIS (subversion revision 7993 at the time), we realized that new functionality helped to solve several everyday problems and opened some possibilities regarding raster data. Among the most prominent features are: the *geography* support that enables distance and area calculations on the ellipsoid, the support for raster data and basic topological errors correction. As of October 2011, PostGIS 2.0 had not been released and was still under heavy development but already featured several improvements which motivated our early adoption. Early tests did not detect any major issue and demonstrated that PostGIS 2.0 was basically backwards compatible with PostGIS 1.5. Some minor bugs where reported by the federal police and quickly corrected by the developers. This work describes the installation requirements, the migration process, the caveats and back up procedures adopted by the federal police in our production database which is online since august 2011. PostGis 2.0 has been released in April 3rd 2012. This work is provided in the hope that it may be useful to anyone considering making the same transition.

1 INTRODUCTION TO POSTGRESQL AND POSTGIS

PostgreSQL is an open source relational database which uses SQL as its query language [6]. It is used by a number of institutions [2] and a large Brazilian bank (Caixa Econômica Federal). In terms of philosophy and function, it compares to Oracle, SQL Server and DB2. It follows a different paradigm from NoSQL, used by other famous databases such as Cassandra, MongoDB and HBase.

PostGIS is an extension for PostgreSQL that defines spatial data types and spatial operations on these data types. The full list of operations and data types is available at [5]. This extension makes it possible to store geometries in a relational database and perform spatial queries alongside regular queries. PostGIS fulfils the same role in PostgreSQL as Oracle Spatial in the Oracle database.

PostgreSQL and PostGIS are free and open source, specific details are covered by their licenses in [4, 3]. They cost no money to download and use, including for commercial purposes, and both run on a variety of platforms including, but not limited to, BSD, Linux, Mac OS X and Windows.

1.1 Installation

The PostGIS documentation includes a detailed description of the installation steps. This section gives a brief overview of the process to give the reader an idea of the work involved. In our experience, installing PostGIS 1.5.3 usually takes well under two minutes of extra time beyond the regular PostgreSQL installation. PostGIS 2.0 is not yet on the mainstream distribution channels ("stack builder" and linux package repositories), and therefore requires an extra step that can take from less than one minute to more than one hour if the user decides to compile from source.

The Windows installation uses a graphical interface and may follow steps similar to these:

1. Download and install PostgreSQL, do not check the option "stack builder".
2. Download and install the PostGIS 2.0 installer from `www.postgis.org`.
3. Create a database instance and enable PostGIS with the command "CREATE EXTENSION postgis;"

The Linux installation involves similar steps:

1. Download and install PostgreSQL using a package manager of your choice.
2. Install PostGIS using one of the following:
   Download a package from an alternate channel (faster)
   Compile from source (more reliable)
3. Create a database instance and enable PostGIS with the command "CREATE EXTENSION postgis;"

After these steps the user can populate the database with data and start using it. Compiling from source may take a long time mainly because the user has to gather appropriate versions of all the necessary libraries and make sure that the resulting environment is sound for entering production. The faster approach is to use an alternate source like *ppa:sharpie/postgis−nightly* (this works for Ubuntu Server), but such sources available at this time are clearly marked as unstable.

PostGIS 2.0 can be installed on an existing database using the same procedures (it is recommended that any changes - related or not to PostGIS - to a production database are first validated on a testing environment).

1.2 Usage

Let us consider the following questions an Environmental Forensics expert in the Brazilian Federal Police might need answered:

- Is the deforested area in a national park?
- How far is the nearest water body from the landfill?
- When was this area deforested?
- Does this person have a mining permit for extracting gold in this part of the river?
- Have any forensic reports ever been produced near this area?

Let us also consider a hypothetical PostgreSQL 9.1/PostGIS 2.0 database containing the tables used in the examples populated with the correct information.

For the first question, the following example query would list all the national parks that intersect with that area and output the actual overlapping area:

```
SELECT name,ST_Intersection(geom,deforested_area)
 FROM protected_areas.national_parks
 WHERE ST_Intersects(geom,deforested_area);
```

For the next question, this is how we would find the distance in meters, the name and the geometry of the N nearest water bodies:

```
SELECT
  name,
  ST_Distance(geom::geography,
   landfill_geom::geography) as distance,
  geom
 FROM physical_maps.water_bodies
 ORDER BY distance ASC NULLS LAST
 LIMIT N;
```

The third question may not be directly answerable using only a query, but we may search the image footprint database for satellite images of the area for visual inspection:

```
SELECT id,date,download_path,geom
 FROM satellite_images.image_catalog
 WHERE ST_Intersects(geom,deforested_area);
```

Another alternative would be to use the new raster extension in PostGIS 2.0 for querying the satellite images directly and returning just the part of the images that overlap the area of interest.

The other two questions could be answered in a similar way. These examples demonstrate what kind of usage a spatially enabled database could be put to. End users would not make this kind of database access, but these queries could be run on the background by an application.

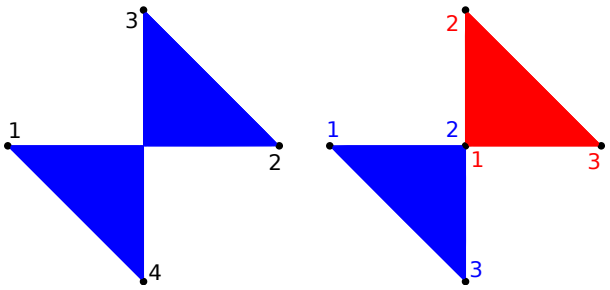
2 POSTGIS 2.0 FEATURE HIGHLIGHTS

PostGIS 2.0 has several new features and improvements from version 1.5.3. Some of these features are very interesting from a database maintenance but these stand out from the rest.

2.1 Fixing invalid geometries

The new function *ST_MakeValid* tries to automatically produce a valid version of invalid geometries without losing vertices. This operation does not work with all possible invalid geometries, but it tries to salvage as much as possible. That functionality is important for Inteligeo because the data in the system is produced by third parties and arrives with very low quality control. *ST_MakeValid* provides an alternative to discarding invalid geometries from the source data.

The new function *ST_IsValidDetail* can be of great help if one tries to correct the data manually because it provides details on the reason for the invalidity and the precise location of the violation. That can help a GIS analyst to be more efficient at correcting inconsistencies.

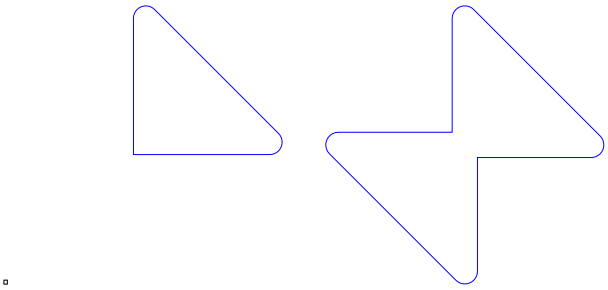


(a) Invalid “bowtie” polygon (b) Topologically correct version

Figure 1: Invalid geometry before and after correction with *ST_MakeValid*

Figure 1a displays the graphical representation of the polygon defined in the WKT form **POLYGON((-1 0, 1 0, 0 1, 0 -1, -1 0))**, which is considered to be topologically invalid (section 4.3.5 of [5]) because it self-intersects. Another issue with this geometry is that vertices have to be traversed in the clockwise order for the boundaries of polygons and counter-clockwise for the holes inside polygons. The implications of this are immediately perceived when we try to calculate the area of such polygon: *ST_Area* of the geometry in Figure 1a yields 0 (zero area). The same operation applied on the geometry generated by *ST_MakeValid* on Figure 1b produces the correct area of 1. That geometry can be represented in WKT as **MULTIPOLYGON(((-1 0, 0 0, 0 -1, -1 0)),((0 0, 0 1, 1 0, 0 0)))**.

Other geometrical operations also produce unexpected results when applied to invalid geometries; Figure 2 displays the result of buffering (*ST_Buffer*) the geometries in Figure 2.



(a) Buffer of the invalid polygon (b) Buffer of the corrected geometry

Figure 2: Results of the operation *ST_Buffer* with a parameter of 0.1

2.2 Improved distance measurement

The regular distance measurement functions make calculations using cartesian trigonometry with the latitude and longitude as coordinates. If the data is in geographical coordinates, such distance measurements would be in degrees and would not correspond to the actual distance in the surface of the earth. For example: the cartesian distance between the two points located at latitude 89 degrees and longitudes -40 and +40 degrees is exactly 80 degrees, which is the same distance for the points in the same longitudes but at a latitude of 0 degrees. In the second case the two points are much farther apart but *ST_Distance* returns the same value.

To get reasonably accurate distances, the geographical coordinates are converted to a projected coordinate system, like the Universal Transverse mercator (UTM), which is divided in 60 zones with a 6^o width each one. Distance measurements of points in different zones is not straightforward and Brazil is covered by 8 such zones.

Postgis 2.0 defined a new data type called *geography*. This data type redefines operations like *ST_Distance* to take into account an ellipsoidal model for the surface of the earth and not a cartesian plane. For operations like *ST_Buffer*, the data is first converted to a projection chosen automatically and then the calculations are performed as in a cartesian plane. This data type can be used in the entirety of Brazil's territory with great ease. Some data layers can not be represented in projected coordinates because they span the whole country.

Figure 3 illustrates the difference between calculations on the cartesian plane and on the ellipsoid. Two buffer polygons have been calculated around the point of coordinates $latitude = -30^o$ and $longitude = -51^o$. The green oval was calculated using the new *geography* data type with a distance of 9648m and has an area about 15% smaller than the red oval, which was calculated using the new *geometry*.

This functionality is still being implemented. Not all operations are supported with *geography*. The postgis documentation mentions limited support for *ST_Buffer* and possible unexpected buffer results if the object falls between two UTM zones or crosses the date line.

2.3 New tools

Figure 4 demonstrates one possible use of this feature to precompute important maps for forensics use, one of them being the special areas arount water bodies which are calculated based on the distance to the water. This kind of computation would be much harder if it were necessary to split the dataset and reproject each part using a different UTM zone and then stitch everything back together.

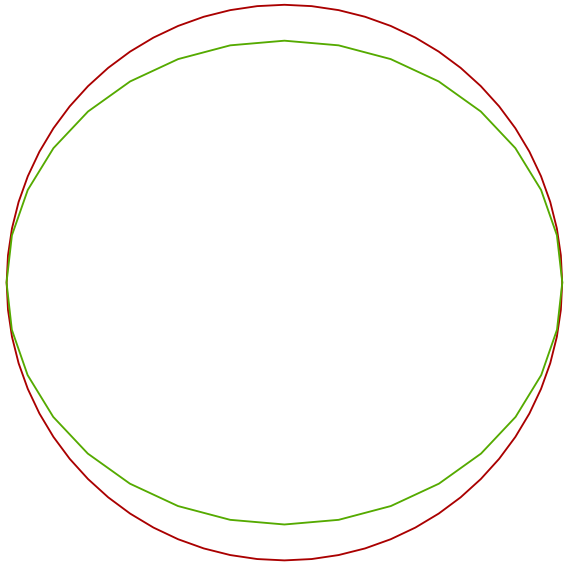


Figure 3: Difference of buffering approaches aroud $latitude = -30^o$ and $longitude = -51^o$. The red line is a buffer of 0.1^o using standard *geometry* data type and the green line is a buffer of 9648m around the same point using the *geography* data type

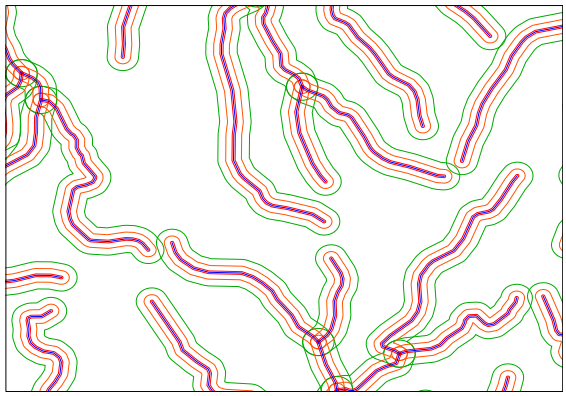


Figure 4: Buffers of 100m (red), 500m (orange) and 1000m (green) around rivers (blue) in the southern region of Brazil

2.4 Use of PostgreSQL Extensions

PostGIS 2.0 can now be installed using postgresql extensions. In practice, this makes installation, backup and upgrading easier.

In the previous version the user had to either use a database template or run two SQL scripts on each database to be spatially enabled, and the scripts were in the host, making it harder to run them remotely without operational system access. With extensions, the user runs the command "CREATE EXTENSION postgis;" needing only a privileged database connection, and everything is in place in about two seconds.

Backup is also simplified. Instead of dumping definitions for each of the postgis functions, the backup only has a few lines defining the extension. An empty database backup goes from about 13,000 lines to about 4,200 lines. The backups are also more portable between postgis versions - restoring an older version backup to a newer version database would just pick up the new definitions of the PostGIS extension instead of gererating conflicting method definitions.

Upgrading a live database is performed by "ALTER EXTENSION postgis UPDATE TO "2.0.1";" (section 2.9.1.2 of [5]). If that fails, a dump/restore procedure is required.

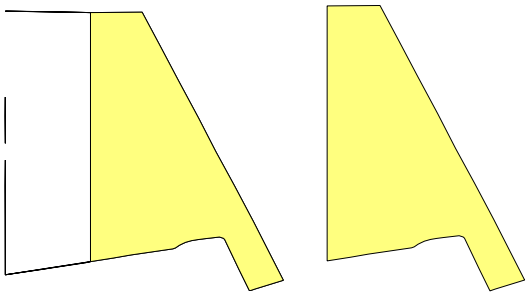
2.5 Support for images

The new raster support permits processing images by using SQL queries. The images can be stored inside the database or "out-db". This has not been tried in Inteligeo yet but it may be a good tool to provide users with only the image part they need. This is important in regions of poor network connectivity.

3 RESULTS

Inteligeo production database was migrated to a development version of PostGIS 2.0 in October 2011 and after 6 months of continuous operation and very few updates, there have been no incidents with the database at all. Our database access is almost entirely composed of reads, so it may constitute a different environment from other deployments.

Invalid geometries of new data are no longer being dropped. Every invalid feature is processed by the ST_Makevalid function.



(a) Geometry before processing (b) Geometry after processing

Figure 5: Spike removal of data before insertion into the database.

Figure 5 is an example of spike removal, one type of processing used in Inteligeo. This procedure relies on buffer operations and can be performed safely only on valid geometries. Figure 5 shows typical artifacts remaining from operations such as the inexact subtraction of one geometry from the other. This is similar to the image processing morphological operations of an *Closing* followed by and *Opening* using radius 0.00001 [1]. The radius of the operations is adjusted according to the dataset, and the 'join=mitre' option is used to preserve as much as possible the original geometry and not round the corners:

```
ST_Buffer(
 ST_Buffer(
  ST_Buffer(ST_MakeValid(geom),
   0.00001, 'join=mitre'),
  -0.00002, 'join=mitre'
  ),
 0.00001, 'join=mitre'
 )
```

3.1 Application compatibility

Postgis 2.0 supports the same queries as version 1.5.3. Our maps are served using ArcGIS 10.0 connected to the database using "query layers" (no ArcSDE is involved). Other than the labor of manually adjusting about 800 layer pointers (ArcGIS 10.0 lacks automatic adjustment of query layer sources), the new database worked well, with no problems. We do most of our quality control using QuantumGis, which also worked fine with the new database, except for Ticket #1200, which the team submitted and the problem got fixed in less than a week.

3.2 Support

The team reported three bugs on the website of the project during this time (Tickets 1197, 1198 and 1200). Two of them were addressed on the same week (1197 and 1200) and I got a personal contact about the third one (1198) telling me that they would wait for more data, since the bug was very difficult to reproduce and may have been due to the GEOS library (not PostGIS).

The community support for this project is very good. If professional support is needed, it is possible to hire the companies that employ some of the PostGIS developers, with the added advantage that they may commit changes to the project code.

3.3 Maintenance

The extension support makes restoring from backup much easier. The main maintenance burden is tracking the source code releases of PostGIS, GEOS and GDAL and recompiling the source code every time an important bug is fixed in any of them. This has happened about 3 times in 6 months. Other than monitoring the releases, there are no significant tasks beyond regular database maintenance - keeping up-to-date backups, monitoring statistics, tuning and making upgrades between minor versions.

ACKNOWLEDGEMENTS

The authors would like to thank the Japan International Cooperation Agency (JICA, `www.jica.go.jp/english`, Alos4Amazon project), the U.N. Office on Drug and Crime (UNODC, `www.unodc.org`) and the national Financier of Projects and Studies (FINEP, `www.finep.gov.br`) for sponsoring the project in which this work is inserted.

References

González, R. and Woods, R., 1992. Digital image processing. Addison-Wesley world student series, Addison-Wesley.

Group, P. G. D., 2012. *PostgreSQL Featured Users*. PostgreSQL Global Development Group. `http://www.postgresql.org/about/users/` (10 Apr. 2012).

Multiple, 1991. *The GNU General Public License (GPL 2.0)*. Free Software Foundation, Inc. `http://www.opensource.org/licenses/gpl-2.0.php` (10 Apr. 2012).

Multiple, 1996. *The PostgreSQL License*. PostgreSQL Global Development Group and The Regents of the University of California. `http://www.postgresql.org/about/licence/` (10 Apr. 2012).

Multiple, 2012a. *PostGIS 2.0.0 Manual*. Refractions Research, Inc. `http://postgis.refractions.net/documentation/manual-2.0/reference.html` (10 Apr. 2012).

Multiple, 2012b. *PostgreSQL 9.1.3 Documentation*. PostgreSQL Global Development Group. `http://www.postgresql.org/docs/9.1/static/intro-whatis.html` (10 Apr. 2012).

Revised April 2012