Ejemplos prácticos para manejar datos raster en PostGIS 2.0.0

Autor: Iván Lizarazo[1]

Fecha: Junio 22 de 2012

1. Introducción

Este documento muestra ejemplos de utilización práctica de algunas funciones que permiten manejar información raster en PostGIS. Los ejemplos han sido probados en una instalación de PostgreSQL 9.1 y PostGIS 2.0.0 en el sistema Windows 7.

Este documento supone que el lector entiende los conceptos básicos de creación de bases de datos, tablas geométricas y consultas SQL en PostGIS.

Antes de iniciar, hay que indicar que este documento se elaboró únicamente con el propósito de familiarizar al lector con las funcionalidades raster presentes en PostGIS 2.0.0. Por lo tanto, no se presentará una descripción detallada de todos los argumentos que una función específica requiere.

Para una mejor comprensión sobre el uso de las diferentes funciones de PostGIS, se sugiere que el lector consulte la documentación disponible en www.postgis.org.

Este documento es distribuido a través de GeoTux Soluciones Geoinformáticas Libres cuya dirección web es http://geotux.tuxfamily.org/

El autor espera que este documento contribuya a promover el uso de PostGIS en el medio académico y empresarial.

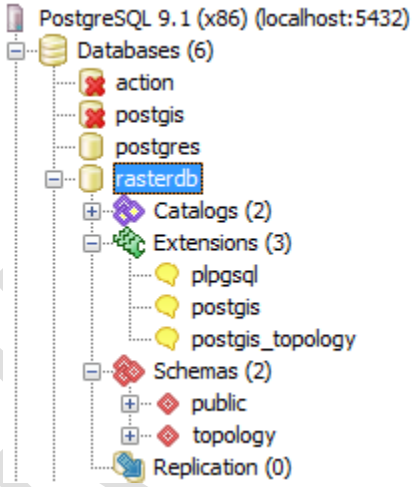
[1] e-mail: *ilizarazo@udistrital.edu.co*

2. Cargue y creación de datos

2.1 Cargue de datos existentes

La herramienta *raster2pgsql* permite convertir datos raster que estén en formatos soportados por GDAL en un archivo sql. Este archivo se puede posteriormente almacenar en una tabla raster de PostGIS. La herramienta *raster2pgsql* permite importar directorios completos lo mismo que crear versiones reducidas (generalizaciones u overviews) de los datos raster.

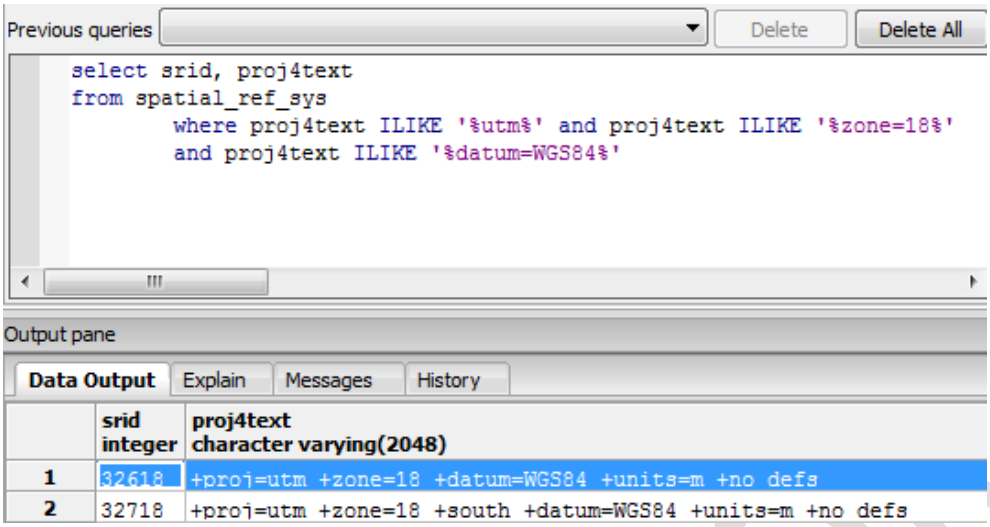
El pre-requisito obvio para importar una imagen a una tabla raster es tener una base de datos PostGIS 2.0. En nuestro caso, primero hemos creado una base de datos llamada *rasterdb* tal como se puede ver en el navegador de objetos de pgAdminIII que se muestra enseguida:



Enseguida se explicará cómo usar la herramienta *raster2pgsql* para importar una imagen Landsat-TM, compuesta por seis bandas espectrales, a la base de datos *rasterdb*. Sabemos que la imagen se encuentra referenciada en la proyección cartográfica UTM Zona 18N cuyo datum geodésico es WGS84. El nombre de la imagen es *recorte.tif* y su tamaño en disco es de 6.15 MB[2]. El Anexo 1 ofrece más detalles sobre esta imagen.

Lo primero que debe hacerse antes de importar una imagen es averiguar cuál es el código que identifica su referencia espacial (mejor conocido como SRID). Para ello, se puede realizar una consulta, desde el editor SQL de pgAdminIII, a la tabla *spatial_ref_sys* de la base de datos *rasterdb.* Tal como se indica en la figura siguiente el SRID de la imagen es 32618.

[2] Esta imagen se puede descargar en este enlace: http://db.tt/b3UWvir4



Una buena práctica al importar datos raster es la de "cortarlo" en piezas (tiles en inglés) que pueden ser, por ejemplo, de 100x100 pixeles. Igualmente, es conveniente utilizar la opción –C para aplicar restricciones y asegurar que el raster tenga un registro apropiado en el catálogo raster, la opción –I para crear un índice GIST de la tabla raster, lo mismo que la opción –M para forzar el análisis de dicha tabla.

La herramienta *raster2pgsql* se debe ejecutar en una ventana de comandos de Windows tal como se indica enseguida:

```
c:\datos\raster>c:/pgutils/raster2pgsql -s 32618 -I -C -M recorte.tif -F -t 100x
100 public.chia > chia.sql
```

Observe, en las instrucciones anteriores, que la imagen *recorte.tif* se encuentra almacenada en el directorio c:\datos\raster y que las herramientas de postgresql están en el directorio c:\pgutils[3].

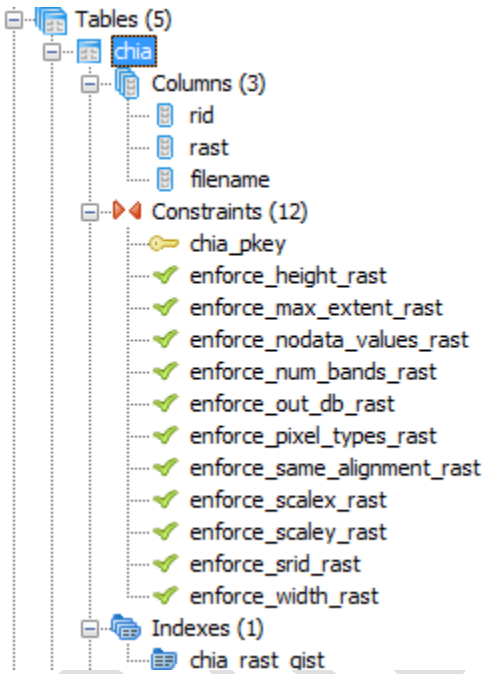
Posteriormente, se debe ejecutar la siguiente instrucción para convertir el archivo *chia.sql* en una tabla raster:

```
c:\datos\raster>c:/pgutils/psql -d rasterdb -U postgres -f chia.sql
```

Observe que el usuario que ha realizado dicha acción es el administrador de la base de datos. Por lo tanto, el sistema requiere que se proporcione la contraseña correspondiente antes de ejecutar la acción solicitada.

[3] Estas herramientas usualmente se encuentran en el directorio *bin* de la instalación de PostgreSQL

Como resultado de la última instrucción, en el esquema público de la base datos *rasterdb* se deberá haber creado una tabla raster de nombre *chia* como se observa en la siguiente ilustración:



Para conocer las características de la tabla raster recién creada se puede ejecutar la siguiente consulta:

Previous queries | Delete | Delete All

```
select count(*) as num_tiles, st_height(rast) as filas,
        st_width(rast) as width, st_srid(rast) as srid,
        st_numbands(rast) as bandas
from chia
group by st_height(rast),
        st_width(rast), st_srid(rast),
        st_numbands(rast);
```

El resultado obtenido es el siguiente:

Output pane

Data Output | Explain | Messages | History

	num_tiles bigint	filas integer	width integer	srid integer	bandas integer
1	90	100	100	32618	6

Observe que la tabla *chia* contiene las seis bandas originales. Esto ocurrió porque no se le indicó a la herramienta *raster2pgsql* ningún parámetro relacionado con ese tema y, por default, se importaron todas las bandas. En

caso que se desee importar solamente algunas bandas hay que utilizar la opción –b e indicar los números de las bandas de interés separados por comas, por ejemplo: *-b 1,2,3 .*

Para visualizar la información almacenada en la tabla chia se puede utilizar el software QGIS. En nuestro caso, se utilizó el plugin *Load Postgis Raster to QGIS* para visualizar la capa. El resultado se puede observar en la siguiente ilustración en la cual se muestra una composición a color RGB321:

2.2 Uso de funciones de mantenimiento de tablas raster

Las tablas raster quedan registradas en el catálogo raster que es una tabla que se llama *raster_columns*. Esta tabla está disponible para consulta como se indica enseguida:

```
select r_table_name as nombretabla, r_raster_column,srid, pixel_types,
        scale_x, scale_y, num_bands, extent
from raster_columns
where r_table_schema = 'public';
```

El resultado obtenido es el siguiente:

Output pane

Data Output | Explain | Messages | History

	nombretabla name	r_raster_column name	srid integer	pixel_types text[]	scale_x double precision	scale_y double precision	num_bands integer	extent geometry
1	otraimagen	rast	0	{8BUI,8BUI	1	-1	4	01030000
2	chia	rast	32618	{8BUI,8BUI	30	-30	6	01030000
3	dem	rast	26904	{16BUI}	10	-10	1	01030000

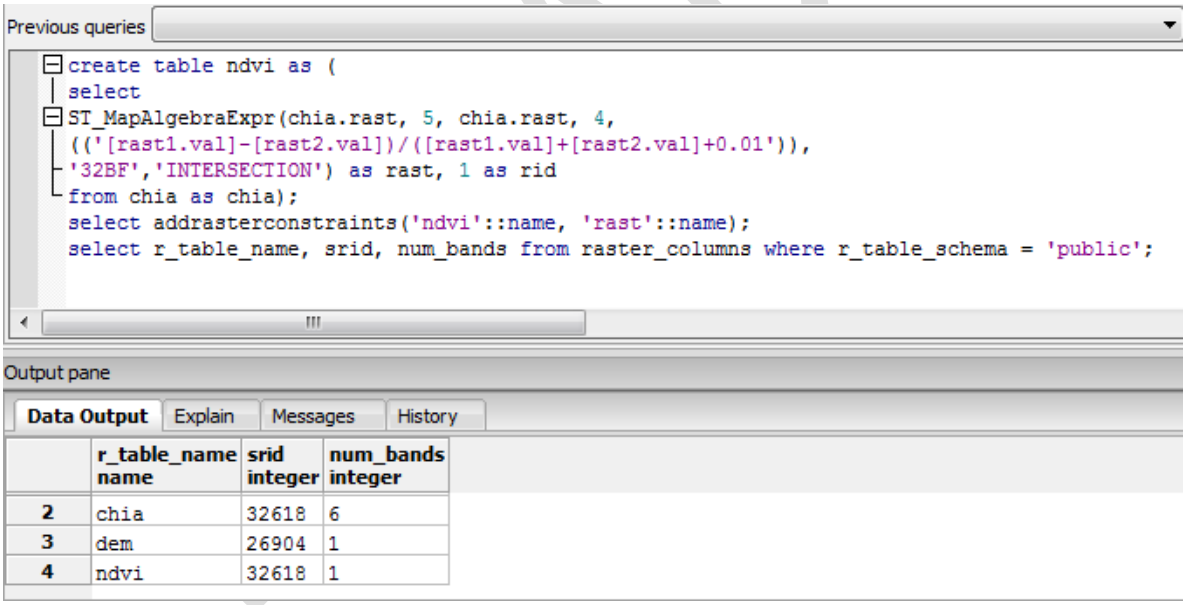
Observe que la tabla raster *chia* está compuesta por 6 bandas cuyos pixeles tienen datos del tipo *8-bit unsigned integer* (8BUI). Observe que el tamaño del pixel de *chia* es de 30 metros.

3. Funciones de análisis raster

3.1 Obtención de nuevas bandas

Es posible obtener nuevas bandas mediante la ejecución de operaciones de algebra de mapas en PostGIS. A manera de ejemplo, se explicará cómo obtener el índice de humedad (NDWI) a partir de la tabla raster *chia*. El NDVI de cada pixel de una imagen se obtiene restando el nivel digital de la banda infrarroja cercana (banda 4 en Landsat-TM) del nivel digital de la banda infrarroja media (banda 5 de Landsat-TM) y dividiendo el resultado por la suma de dichos niveles[4].

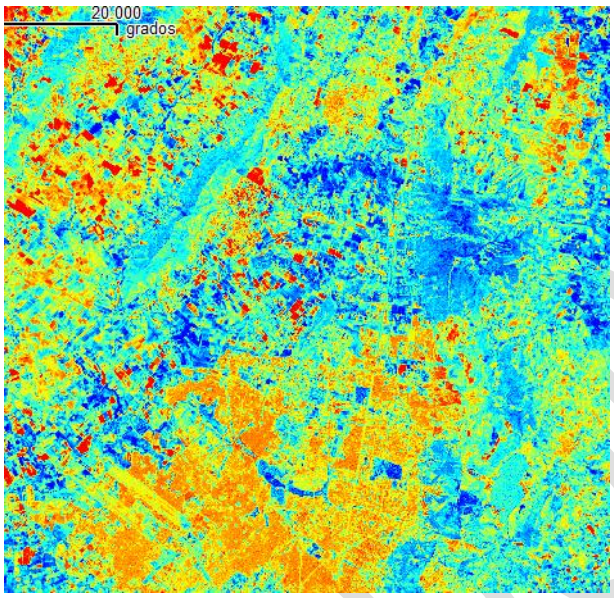
La siguiente instrucción SQL permite obtener una tabla raster de nombre *ndvi* en donde se almacenan los valores de NDWI calculados mediante una expresión de algebra de mapas:



Observe que la función que realiza el cálculo es ST_MapAlgebraExpr. El resultado de la operación se almacena en el tipo de datos *32-bit float* (32BF). Observe también que se ha utilizado una instrucción para agregar restricciones raster a la tabla NDVI. Ello asegura que el catálogo *raster_columns* registre las propiedades de la tabla recién creada.

[4] Este índice se debería calcular usando valores de reflectancia de superficie en lugar de niveles digitales originales como se indica aquí. Pero esa historia está fuera del alcance de este documento.

La siguiente ilustración muestra el resultado de visualizar la tabla raster *ndvi* usando QGIS:



Un repaso rápido a los metadatos de esta imagen, desde QGIS, permite comprobar que esta tabla consta de una sola banda y que los valores NDWI se encuentran dentro del rango -0.55 a 0.65. Los valores más altos corresponden a las zonas de mayor humedad.

3.2 Extracción de valores de un raster usando geometrías

Enseguida vamos a mostrar cómo obtener los valores de NDVI a lo largo de un transecto de interés. Para el efecto, crearemos una tabla que almacene dicho transecto y le agregaremos dos registros:

Previous queries | Delete | Delete All

```
-- creación de la tabla
create table transectos (
gid serial PRIMARY KEY,
cobertura char(10)
);
-- adición de geometría a la tabla
select addgeometrycolumn('transectos','geom', 32618, 'LINESTRING',2);
-- inserción de datos a la tabla
insert into transectos (gid, geom,cobertura) values
(1, st_linefromtext('LINESTRING(593100 536100, 593500 536300)',32618),'cultivos');
insert into transectos (gid, geom,cobertura) values
(2, st_linefromtext('LINESTRING(612000 519500, 612000 521000)',32618),'agua');
-- consulta de la tabla
select gid, st_asewkt(geom), cobertura from transectos;
```

El resultado de la consulta realizada a la tabla *transectos* es el siguiente:

Output pane

Data Output | Explain | Messages | History

	gid integer	st_asewkt text	cobertura character(10)
1	1	SRID=32618;LINESTRING(593100 536100,593500 536300)	cultivos
2	2	SRID=32618;LINESTRING(612000 519500,612000 521000)	agua

La siguiente ilustración muestra los dos registros de la tabla *transectos* desplegados sobre una composición RGB321 de la tabla *chia*, visualizados en QGIS. A la izquierda aparece el transecto correspondiente a cultivos y a la derecha el correspondiente a agua:

La siguiente instrucción permite obtener el valor raster de la tabla ndvi en el centroide de cada uno de los transectos:

```
SELECT transectos.gid, transectos.cobertura,
st_value(ndvi.rast, st_centroid(transectos.geom),FALSE) as ndvi_value
FROM transectos, ndvi
where st_intersects(ndvi.rast, transectos.geom);
```

El resultado es el siguiente:

	gid integer	cobertura character(10)	ndvi_value double precision
1	1	cultivos	-0.216970890760422
2	2	agua	
3	2	agua	0.0434593670070171

También es posible obtener los valores raster de las bandas de interés. En la siguiente consulta vamos a indagar por los valores de las bandas 4 y 5 asociados al centroide de cada uno de los transectos:

```
SELECT transectos.gid, transectos.cobertura,
st_value(chia.rast,4, st_centroid(transectos.geom),TRUE) as band4,
st_value(chia.rast,5, st_centroid(transectos.geom),TRUE) as band5
FROM transectos, chia
where st_intersects(chia.rast, transectos.geom);
```

El resultado es el siguiente:

	gid integer	cobertura character(10)	band4 double precision	band5 double precision
1	1	cultivos	129	83
2	2	agua		
3	2	agua	11	12

Es posible que usted esté pensando, con razón, que no es suficiente con obtener los valores raster en el centroide del transecto. Bueno, las instrucciones para obtener dichos valores a lo largo de cada uno de los puntos que constituyen el transecto correspondiente a la cobertura de cultivos es la siguiente:

```
-- cálculo de valores raster a lo largo del transecto
SELECT tr.gid, tr.cobertura, st_value(ndvi.rast, st_centroid(
(st_intersection(ndvi.rast, tr.geom)).geom), FALSE) as ndvi_value
FROM transectos as tr, ndvi as ndvi
where st_intersects(ndvi.rast, tr.geom) and tr.cobertura ='cultivos';
```

Una pregunta obvia respecto a la instrucción anterior es por qué hay necesidad de obtener el centroide de la geometría que resulta de intersectar el transecto vectorial y la capa raster. Bueno, la razón es que dicha intersección no es una serie de puntos, como pareciera lógico, sino segmentos de línea. Segmentos de línea? Sí, tal como lo leyó. Enseguida volveremos a ese tema.

El resultado de la consulta indicada en la página anterior es el siguiente:

	gid integer	cobertura character(10)	ndvi_value double precision
1	1	cultivos	-0.203873604536057
2	1	cultivos	-0.299525380134583
3	1	cultivos	-0.245271444320679
4	1	cultivos	-0.141456514596939
5	1	cultivos	-0.282033205032349
6	1	cultivos	-0.322015702724457
7	1	cultivos	-0.295760750770569
8	1	cultivos	-0.244007468223572
9	1	cultivos	-0.249988421797752
10	1	cultivos	-0.273669809103012
11	1	cultivos	-0.311613410711288
12	1	cultivos	-0.276982307434082
13	1	cultivos	-0.216970890760422
14	1	cultivos	-0.309076845645905
15	1	cultivos	-0.31505411863327
16	1	cultivos	-0.280360728502274
17	1	cultivos	-0.286417782306671
18	1	cultivos	-0.290309190750122
19	1	cultivos	-0.291852056980133
20	1	cultivos	-0.271831452846527
21	1	cultivos	-0.292439043521881

Regresando al tema de cuál es la geometría de que resulta de intersectar el transecto vectorial y la capa raster, la mejor manera de resolver la inquietud es preguntándole directamente a PostGIS:

```
-- que tipo de geometria es la interseccion entre un raster y una línea
SELECT tr.gid, tr.cobertura,
st_astext((st_intersection(ndvi.rast, tr.geom)).geom)
as geometria,
st_value(ndvi.rast, st_centroid(
(st_intersection(ndvi.rast, tr.geom)).geom), FALSE) as ndvi_value
FROM transectos as tr, ndvi as ndvi
where st_intersects(ndvi.rast, tr.geom) and tr.cobertura ='agua';
```

El resultado es el siguiente:

	gid integer	cobertura character(10)	geometria text	ndvi_value double precision
1	2	agua	LINESTRING(612000 520965,612000 521000)	0
2	2	agua	LINESTRING(612000 520875,612000 520965)	0.0434593670070171
3	2	agua	LINESTRING(612000 520845,612000 520875)	0
4	2	agua	LINESTRING(612000 520815,612000 520845)	0.0832986235618591
5	2	agua	LINESTRING(612000 520755,612000 520785)	0

Cuál es la explicación de ello? Antes de realizar la intersección, PostGIS convierte las celdas raster en polígonos. Por consiguiente, al intersectar dichos

polígonos con un transecto se obtiene como resultado una serie de segmentos de línea. Por otro lado, la función ST_Value retorna el valor de una banda específica en un punto geométrico. Ello significa que hay que pasarle como argumento una geometría de punto y no una de línea. Por consiguiente, en la consulta correspondiente se hace necesario obtener primero el centroide de cada segmento de línea que conforma la intersección. Seguramente que habrá otras soluciones pero la indicado aquí es funcional.

4. A manera de conclusión

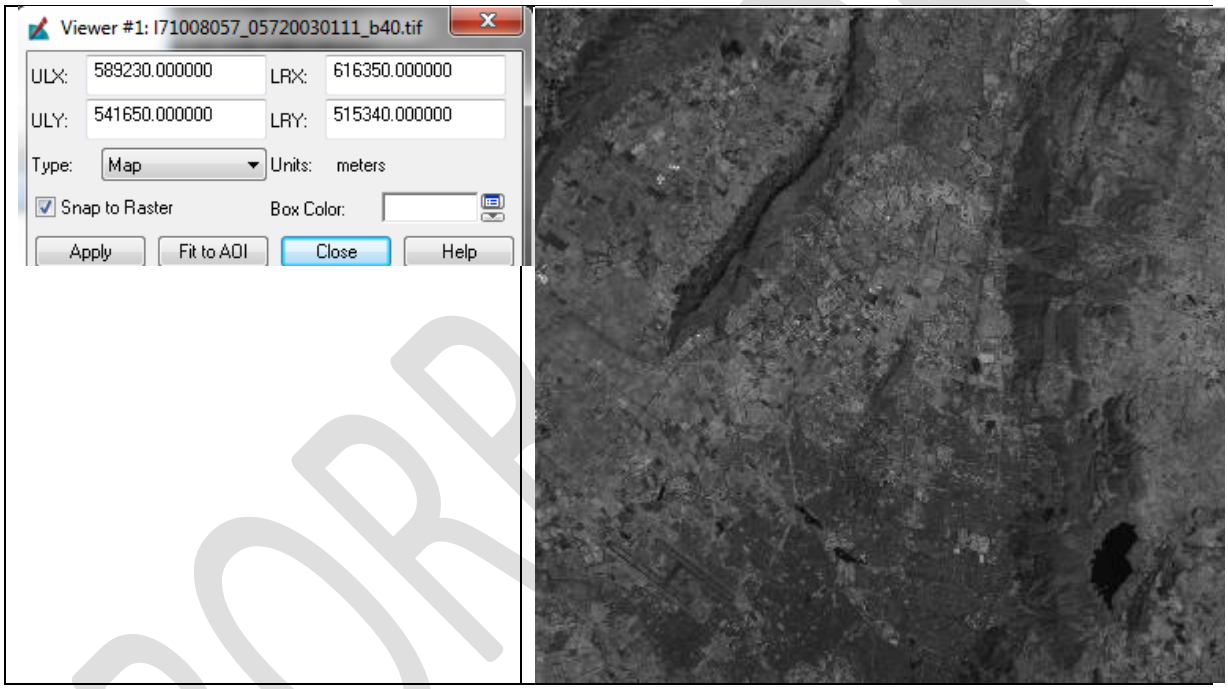
Este documento mostró algunos ejemplos prácticos sobre el manejo de datos raster en PostGIS. El tema es muy extenso y esta fue apenas una introducción al mismo. Si usted desarrolla ejemplos adicionales tomando como base este documento, tal vez decida compartirlos. En tal caso, puede escribirle al autor para que su contribución aparezca en la próxima versión de este documento.

5. Referencias

PostGIS Project, 2012. *PostGIS 2.0.0 manual SVN Revision (9605)*. Disponible en línea en http://postgis.org/documentation/manual-2.0/ . Ultimo acceso: 22/06/2012.

Anexo 1

La imagen importada corresponde a un recorte de una escena completa de una imagen Landsat-TM que consta de seis bandas multiespectrales (no incluye la banda térmica). En la parte izquierda de la siguiente ilustración se indican las coordenadas utilizadas para recortar la imagen. En la parte derecha se muestra la banda 1 de la imagen recortada. El recorte corresponde a una zona de la Sabana de Bogotá en el centro de Colombia. La altitud promedio es de 2600 m sobre el nivel del mar. El relieve comprende terreno plano y unas pocas zonas de terreno muy inclinado.



Anexo 2

La siguiente instrucción permite obtener una tabla raster vacía que tiene las mismas características geométricas de la imagen *chia*, es decir número de columnas (905), número de filas (878), coordenadas del origen (situado en la esquina superior izquierda), tamaño de pixel en *y/x* y valor SRID:

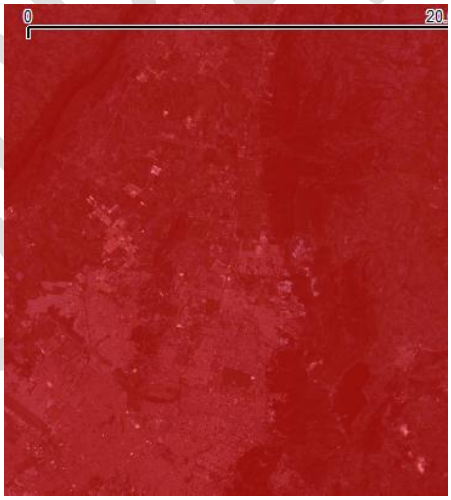
```
-- creacion de una tabla raster
CREATE TABLE dummy_chia(rid integer PRIMARY KEY, rast raster);

-- creacion de un raster vacío en función de parámetros
INSERT INTO dummy_chia(rid, rast)
VALUES (1, st_makeemptyraster (905,878, 589215,541665,30,-30,0,0,32618));

-- agregar una banda de tipo de datos 8BUI con pixeles inicializados en 150
update dummy_chia
       set rast = st_addband(rast,1, '8BUI', 150)
where rid=1;

select addrasterconstraints('dummy_chia'::name, 'rast'::name);
```

La siguiente ilustración muestra la visualización de la nueva tabla raster *dummy_chia* en QGIS:



Observe que la tabla *dummy_chia* se muestra en un color rojo y con un nivel de transparencia del 50%. Debajo de dicha tabla se alcanza a observar la tabla *chia* la cual está desplegada en una composición RGB321.

Anexo 3

El sitio ase-research.org/basille/pg describe funciones útiles para interactuar entre R y PostGIS. Las funciones necesitan el paquete RODBC. Todavía no he probado estas funciones.