

Entregable Talleres
PgRouting, Funciones y triggers

Estudiante: Gabriela Chamorro
Código:2020
Docente:Fabio Andrés Herrera

Universidad del Valle
Especialización en Geomática
Lenguajes de Marcado para SIG en WEB
Santiago de Cali, abril de 2021

RESUMEN

El desarrollo del presente taller consiste en primera instancia en una fase teórica, que como su nombre lo indica, incluyen los conceptos básicos, impartidos en clase por el docente de acuerdo al tema en particular. Con base en esto, se procede al desarrollo de los ejercicios que relacionan los temas de PgRouting, Funciones y Triggers. Adicionalmente, fue necesario emplear el editor de texto Notepad++, el gestor de base de datos PgAdmin con la extensión PostgreSQL, PostGIS y Pgrouting, además del VirtualBox osgeolive-13.0-amd64.vmdk

METODOLOGÍA Y DESARROLLO

Practica 1: PgRouting

Esta práctica relaciona una serie de consultas espaciales mediante el lenguaje sql, las cuales incluyen la extensión de Pgrouting y algunos comandos como el "CREATE OR REPLACE VIEW", "DELETE", "UPDATE", "WHERE", "ORDER BY" entre otros elementos de Postgis para el geoprocetamiento y análisis de información geográfica.

Para dar inicio, fue necesario previamente la instalación del osgeolive que funciona como máquina virtual basada en Linux, la cual soporta gran variedad de softwares espaciales de código abierto para almacenamiento, edición, visualización, análisis y manipulación de datos espaciales.



Instalado el OSGeoLive, se realiza el proceso de descarga de la información en formato shape de: Cuerposagua_univalle, Edificios_univalle, Redpeatonal_univalle, Redpeatonal_univalle_vertices, Ríos_univalle, Sitiosinteres_univalle, Ususuelo_univalle y Vías_univalle, y se procede a convertirlos en formato .sql, usando la terminal de comandos (LXTerminal) y la siguiente línea de código:

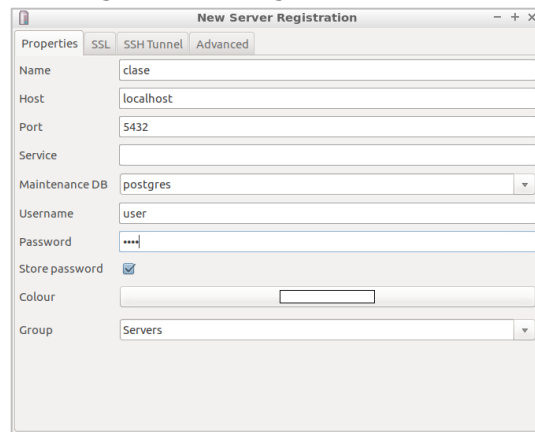
```
shp2pgsql -g the_geom -W LATIN1 -s 4326 redpeatonal_univalle > redpeatonal_univalle.sql
```

Donde:

- shp2pgsql: convierte de shape a sql

- -g the_geom: almacena la componente geográfica dentro del BD
- -W LATIN1: identifica los caracteres especiales en la BD
- -s 4326: define el sistema coordenadas geográficas
- redpeatonal_univalle: nombre de la capa de entrada
- redpeatonal_univalle: nombre de la capa dentro de la BD
- redpeatonal_univalle.sql: nombre de la capa de salida

Se emplea el administrador de bases de datos PgAdmin contenido en la máquina virtual para crear la base de datos denominada "uvruteo", añadiendo la extensión de Postgis, postgresql y pgrouting, teniendo presente la siguiente configuración:



En seguida, se procede a importar cada capa SQL a la base de datos de esquema público a partir de los parámetros descritos a continuación:

```
psql -h localhost -p 5432 -d uvruteo -U user -f redpeatonal_univalle.sql
```

Donde:

- psql importar sql a la bd
- -h el host
- -p el puerto
- -d el nombre de la base de datos
- -U el usuario
- -f el file o archivo sql

Luego se prepara la capa de redpeatonaunivalle para soportar las operaciones de cálculo de ruta. Para ello, a partir de sentencias sql se altera la tabla creando los campos de X1, Y1, X2, Y2, todos ellos de tipo "doblé precisión". En seguida, se realiza el cálculo de las coordenadas de inicio y fin de cada una de los puntos de la red peatonal mediante la siguiente expresión:

```
UPDATE redpeatonal_univalle set x1 = st_x(st_pointn(st_linemerge(the_geom),1));  
UPDATE redpeatonal_univalle set y1 = st_y(st_pointn(st_linemerge(the_geom),1));
```

Así mismo, se crea la topología de la red y los índices espaciales que aceleran la búsqueda al organizar los datos en forma de árbol para realizar rápidamente la búsqueda de un registro en particular. Estos índices se calculan sobre los campos Source y Target tipo "integer" previamente creados.

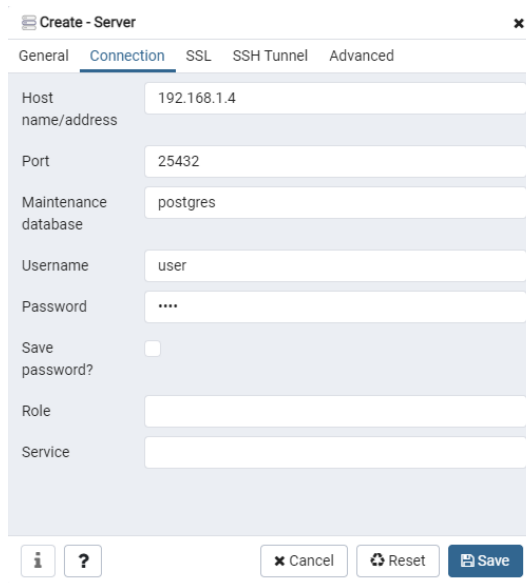
```
SELECT pgr_createTopology('redpeatonal_univalle',0.00001, 'the_geom', 'gid');  
CREATE INDEX ways_source_idx ON redpeatonal_univalle("source");  
CREATE INDEX ways_target_idx ON redpeatonal_univalle("target");
```

También se asigna un costo o criterio de distancia para cada nodo de la red mediante siguiente línea de comando sobre el campo Costo de tipo "doblé precisión" creado en la tabla de redpersonalunivalle.

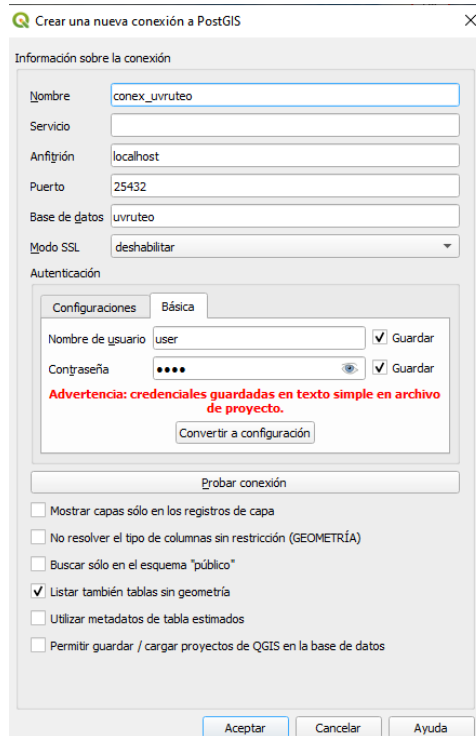
```
UPDATE redpeatonal_univalle SET costo = st_length(st_transform(the_geom,3115));
```

A partir de la creación de las coordenadas y los campos souce, target y el costo se procede a desarrollar las consultas sql para lo cual, se realiza la conexión previa entre el host y la máquina virtual de tal manera que sea posible ejecutar las consultas desde el pgAdmin local y visualizar los resultados en el Qgis así:

- Desde el pgAdmin local se crea la conexión al servidor de la máquina virtual desde "server", en donde se introduce el host (localhost) que corresponde a la dirección IP del PC, el número del puerto de la máquina virtual (25432), seguido de la base de datos (postgres), el nombre del usuario (user) y la contraseña (user)



- Desde el Qgis también se realiza la conexión a la GDB de la máquina virtual bajo los parámetros de nombre (cualquier, ej: conex_uvruteo), Admin (localhost), puerto (25432), base de datos (uvruteo), usuario (user) y contraseña (user) así, las consultas que se ejecuten en el pgAdmin se reflejarán en el espacio de trabajo del Qgis.



Una vez realizada la configuración de la conexión a la base de datos, se procede a desarrollar cada una de las consultas propuestas para el presente taller usando las herramientas de geo procesamiento en "Postgis". Para todas las consultas se crea o reemplaza una vista (VIEW) bajo la siguiente estructura, teniendo en cuenta el un nodo de inicio y un nodo de fin. Luego, dado que función `pgr_dijkstra()` retorna el listado de gid de las líneas que conforman la ruta más corta, se debe realizar un join usando " left join" entre el id de las líneas, y la tabla completa de las líneas para obtener la geometría y guardarla en una vista.

```
CREATE OR REPLACE VIEW resultado_ruteo AS
SELECT seq, id1 AS node, id2 AS edge, cost, b.the_geom FROM pgr_dijkstra('SELECT gid AS id,
source::integer, target::integer,
costo::double precision AS cost
FROM redpeatonal_univalle',1114,351, false, false) a LEFT JOIN redpeatonal_univalle b ON
(a.id2 = b.gid);
```

Dado que para el cálculo de ruta más corta se tiene en cuenta la anterior sintaxis, se estructuran a continuación los siguientes ejercicios, en donde según las condiciones de cada caso solo se ajustan los parámetros del nodo de inicio y fin así:

3. Realizar calculo ruta más corta entre punto inicial dado por coordenadas (lat,lon) y un nodo.

Dentro de la sintaxis base para el cálculo de la ruta más corta, se crea el punto inicial con "st_makepoint" a este se le define el sistema de referencia 4326 usando el comando "ST_SetSRID" y posteriormente se calcula la distancia de éste punto hacia cada uno de los nodos de la capa de red peatonal univalle vértices PGR a través de "st_distance", se ordena la distancia en forma ascendente y se selecciona el primer registro que es el más corto con "limit1". En este sentido se muestra como ejemplo el siguiente código completo:

```
CREATE OR REPLACE VIEW punto3 AS
SELECT seq, id1 AS node, id2 AS edge, cost, b.the_geom from pgr_dijkstra('
select gid as id,
source::integer,
target::integer,
costo::double precision as cost
from redpeatonal_univalle',
(select corto.id::integer from
  (select nodos.id,
st_distance (ST_SetSRID (st_makepoint(-76.53485,3.37275),4326),nodos.the_geom) as
distancia
  from redpeatonal_univalle_vertices_pgr as nodos
  order by distancia asc limit 1) as corto
),1788,false,false) a left join redpeatonal_univalle b on (a.id2 = b.gid);
```

4. Realizar calculo ruta más corta entre punto inicial dado por coordenadas (lat,lon) y punto final dado por coordenadas (lat,lon)

Dentro de la sintaxis base para el cálculo de la ruta más corta, se crea un par de nodos con coordenadas latitud y longitud, se repite el proceso mencionado anteriormente tanto para el nodo de inicio como para el nodo final. (Ver código completo en Taller 1 pgrouting.sql)

5. Realizar calculo ruta más corta entre punto inicial dado por coordenadas (X,Y) y punto final dado por coordenadas (lat,lon). Nota: x,y en 3115

Dentro de la sintaxis base para el cálculo de la ruta más corta, se crea el primer nodo con coordenadas planas al cual se le define el sistema de referencia (3115) con ST_SetSRID" y luego se le aplica una transformación para llevarlo al sistema de referencia geográfica (4326) y el segundo nodo en coordenadas latitud y longitud. (Ver código completo en Taller 1 pgrouting.sql)

6. Para realizar calculo entre un punto inicial dado por coordenadas (lat,lon) y un sitio de interés (buscar el nodo más cercano)

Dentro de la sintaxis base para el cálculo de la ruta más corta, se crea un primer nodo en coordenadas 4326 y el segundo nodo es creado a partir de la tabla de sitiosinterés_univalle, seleccionando como sitio de interés el punto de comidas rápidas, así: sitiosinteres_univalle where name like 'Comidas%'. (Ver código completo en Taller 1 pgrouting.sql)

7. Calcular la ruta más corta entre el CAI y la plazoleta de ingeniería

Dentro de la sintaxis base para el cálculo de la ruta más corta, se crea como nodo 1, desde la capa de sitios de interés el punto de 'CAI%', y como nodo 2 el sitio de interés de 'plazoleta de ingenier%' como punto 2 y se calcula la distancia entre ellos. (Ver código completo en Taller 1 pgrouting.sql)

8. Calcular la ruta entre el punto -76.53427,3.37408 y el punto -76.53231,3.37677. b) Incrementar en un 10% el costo de los segmentos de red peatonal que se encuentran a un radio de 120 metros del punto (-76.53131,3.37446).

A partir de dos pares de coordenadas ubicadas casi que en línea recta y seleccionadas de forma aleatoria, se calcula la distancia más corta entre ellas. Luego mediante un UPDATE se altera el costo en la capa de red peatonal aumentando en un 10%. En seguida se crea la vista para representar el buffer a 120 m del punto donde fue alterado el costo, de tal manera que al recalcular la distancia entre los dos nodos iniciales se observa que la ruta es diferente. Finalmente, para reversar los cambios se aplica un UPDATE en la capa de red peatonal_univalle calculando de nuevo el costo. (Ver código completo en Taller 1 pgrouting.sql).

Práctica 2. Funciones y Triggers

Para ésta práctica se usa la información de la base de datos "UVRUTEO" creada en la práctica anterior y una vez realizada la configuración de la conexión a ésta BD, se procede a desarrollar cada una de las consultas propuestas, el cual se divide el taller en dos partes, la primera hace referencia a consultas de funciones y la segunda a un ejercicio de trigger, ambos se describen a continuación:

Funciones

1. Construir una función para verificar si una palabra es un palíndromo

Se crea la función "EsPalindromo" haciendo uso de las funciones SQL REPLACE que lo que hace es encontrar una cadena de patrón coincidente y reemplazarla por otra cadena, la función LOWER que devuelve una cadena de texto transformada en letras minúsculas y la función REVERSE que se utiliza para invertir la expresión de entrada. En este sentido, al ingresar una frase o palabra, la función comparará el orden del texto respecto a su inverso, si ambos son coincidentes, será una palabra Palíndromo.

```
CREATE OR REPLACE FUNCTION ESPalindromo(Texto text) RETURNS text AS $$
DECLARE
BEGIN
    IF LOWER(REPLACE(Texto, ' ','')) = LOWER(REVERSE(REPLACE(Texto, ' ',''))) THEN
        RETURN 'La palabra: ' || Texto || ' ES un Palíndromo';
    ELSE
        RETURN 'La palabra: ' || Texto || ' NO es un Palíndromo';
    END IF;
END;
$$ LANGUAGE plpgsql
```

2. Construir función que permita convertir coordenadas DMS a DEG.

Se crea la función "convertir_DMSaDEG" con tres parámetros de entrada tipo double precisión, la cual retornará un valor de tipo double precisión, se declaran los parámetros de grados, minutos y segundos y en el cuerpo de la función se configura la instrucción que está asociado a un mensaje o aviso de 'Verifique los parámetros ingresados' en caso de que se ingrese un valor en grados mayor a 90, los minutos mayores a 60 o los segundos mayores a 60. En este sentido se obtuvo lo siguiente:

```
CREATE OR REPLACE FUNCTION convertir_DMSaDEG(double precision, double precision,
double precision)
RETURNS double precision AS $$ /*Tipo de datos que retornará*/
```

```
DECLARE /*Parametros*/
    D ALIAS FOR $1;
    M ALIAS FOR $2;
    S ALIAS FOR $3;
BEGIN /*codigo funion*/
    if D>90 or M>60 or S>60 then raise notice 'Verifique los parámetros ingresados';
    else return D+M/60+S/3600::double precision;
    end if;
END
$$ LANGUAGE plpgsql;
```

3. Construir función que permita calcular ruta más corta entre 2 nodos.

Se crea la función RutaCortaNodos con tres parámetros de entrada, dos de tipo entero que corresponden a los nodos de inicio y fin, adicional a un texto. Luego dentro del Begin se usa "EXECUTE" que se utiliza para ejecutar un comando SQL y asociar el resultado a objetos 4D (arrays, variables o campos) en este caso le asocia la vista "rutaNodosCorta". Este ejercicio es similar a los realizados en el taller pasado, sin embargo, se puede consultar el código completo en "Taller_Triggers.sql".

4. Construir función que permita calcular ruta más corta entre 2 coordenadas

Se crea la función RutaCortaCoordenadas con 4 parámetros de entrada de tipo double que son los pares de coordenadas latitud longitud del punto inicial y final y uno adicional tipo texto. Luego dentro del Begin se usa "EXECUTE" que se utiliza para ejecutar un comando SQL y asociar el resultado a objetos 4D (arrays, variables o campos) en este caso le asocia todos los parámetros para el cálculo de la ruta más corta entre coordenadas. Este ejercicio es similar a los realizados en el taller pasado, sin embargo, se puede consultar el código completo en "Taller_Triggers.sql".

Triggers

Crear un Trigger que permita en una nueva tabla calcular a partir de en un nuevo punto (geometría), el Identificador, nombre y distancia al edificio más cercano, el Identificador y la distancia al sitio de interés más cercano, el identificador y la distancia al sitio de interés más lejano, el azimut comprendido entre el sitio de interés más cercano y el más lejano, el punto medio (geometría) comprendido entre el sitio de interés más lejano y el más cercano, las coordenadas (en 3115) del punto medio comprendido entre el sitio de interés más lejano y el más cercano y la distancia euclidiana entre el sitio de interés más cercano y el más lejano. En este sentido, se creó ejecutan los siguientes pasos:

1. Se creó una tabla llamada "información_edificio" con los campos requeridos

```
DROP TABLE IF EXISTS informacion_edificio;
CREATE TABLE informacion_edificio
    (gid serial PRIMARY KEY,
     x double precision,
     y double precision,
     edificercano_id varchar(150),
     edificercano_nombre varchar(150),
     edificercano_dist double precision,
     sitiocercano_id varchar(150),
     sitiocercano_dist double precisión,
     sitiolejano_id varchar(150),
```



```
sitiolejano_dist double precision,  
azimut double precision,  
puntomedio geometry (POINT,4326),  
pm_x double precision,  
pm_y double precision,  
distancia double precision,  
the_geom geometry (POINT,4326));
```

2. Construcción de las consultas en SQL.
3. Luego, se crea la función llamada "funcion_taller2_trigger()", en la cual dentro de elemento BEGIN se crean lo nuevos elementos (NEW) que corresponden a las consultas estructuradas previamente, así:

```
CREATE OR REPLACE FUNCTION funcion_taller2_trigger()  
RETURNS TRIGGER AS $$  
BEGIN
```

- Se crea el punto x,y

```
NEW.x= st_x(the_geom);  
NEW.y=st_y(the_geom);
```
- El Identificador, nombre y distancia al edificio más cercano.

```
NEW.edifcercano_id =  
(select a.osm_id from  
(SELECT osm_id, name,  
st_distance(the_geom,st_setsrid(st_makepoint(-  
76.5675,3.37263),4326)) as dist  
FROM edificios_univalle ORDER BY dist ASC LIMIT 1) as a);  
  
NEW.edifcercano_nombre=  
(select b.name from  
(SELECT osm_id, name,  
st_distance(the_geom,st_setsrid(st_makepoint(-  
76.5675,3.37263),4326)) as dist  
FROM edificios_univalle ORDER BY dist ASC LIMIT 1) as b);  
  
NEW.edifcercano_dist=  
(select c.dist from  
(SELECT osm_id, name,  
st_distance(the_geom,st_setsrid(st_makepoint(-  
76.5675,3.37263),4326)) as dist  
FROM edificios_univalle ORDER BY dist ASC LIMIT 1) as c);
```
- El Identificador y la distancia al sitio de interés más cercano: igual que el punto anterior solo que se tiene en cuenta la capa de sitios de interés y el elemento ASC LIMIT 1 que indica el menor de la lista (Ver código completo Taller_Triggers.sql)
- El Identificador y la distancia al sitio de interés más lejano: igual que el punto anterior, se tiene en cuenta la capa de sitios de interés y el elemento DESC LIMIT 1 que indica el mayor el valor más alto de la lista. (Ver código completo Taller_Triggers.sql)
- El azimut comprendido entre el sitio de interés más cercano y el más lejano. Se usa la funcion que ofrece el postgres "st_azimuth" que requiere de dos parámetros de entrada

```
NEW.azimut=
(select degrees (st_azimuth(
(select b.the_geom from
(SELECT osm_id,the_geom, name,
st_distance(st_transform(the_geom,3115),st_transform(st_setsrid(st_makepoint(-
76.5675,3.37263),4326),3115)) as dist
FROM sitiosinteres_univalle ORDER BY dist ASC LIMIT 1) as b),
(select b.the_geom from
(SELECT osm_id, name, the_geom,
st_distance(st_transform(the_geom,3115),st_transform(st_setsrid(st_makepoint(-
76.5675,3.37263),4326),3115)) as dist
FROM sitiosinteres_univalle ORDER BY dist DESC LIMIT 1) as b)))));
```

- Las coordenadas (en 3115) del punto medio comprendido entre el sitio de interés más lejano y el más cercano. Se crean dos pares de puntos X,Y , se realiza la transformación al sistema 3115 y se calcula el punto medio entre ellos con ST_distance dividido en 2. (Ver código completo Taller_Triggers.sql)
 - La distancia euclidiana entre el sitio de interés más cercano y el más lejano. Se usa la función st_distance y la capa de sitios de interés_univalle y se realiza la transformación a 3115 para dar el valor de la distancia. (Ver código completo Taller_Triggers.sql)
4. En el último paso se crea el trigger llamado "trigger_taller2" el cual después de insertar o actualizar un elemento en la tabla informacion_edificio, ejecutará la funcion_taller2_trigger();

```
CREATE TRIGGER trigger_taller2
BEFORE INSERT OR UPDATE
ON informacion_edificio
FOR EACH ROW
```

RESULTADOS

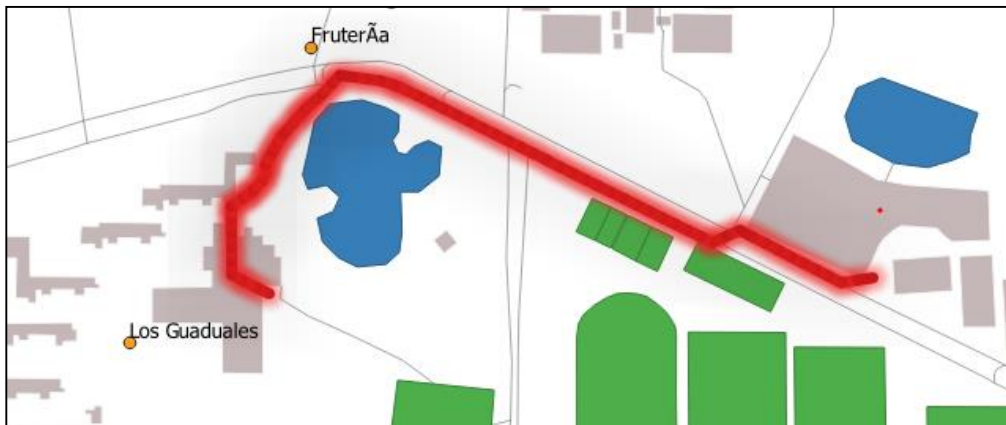
Práctica1

A continuación se muestran los resultados de las vistas creadas para calcular la ruta mas corta, teniendo presente las condiciones planteadas en cada item. Se obtuvieron 9 capas cargadas en la base de datos Uvruteo instalada en el pgAdmin y en sistema de coordenadas planas 3115.

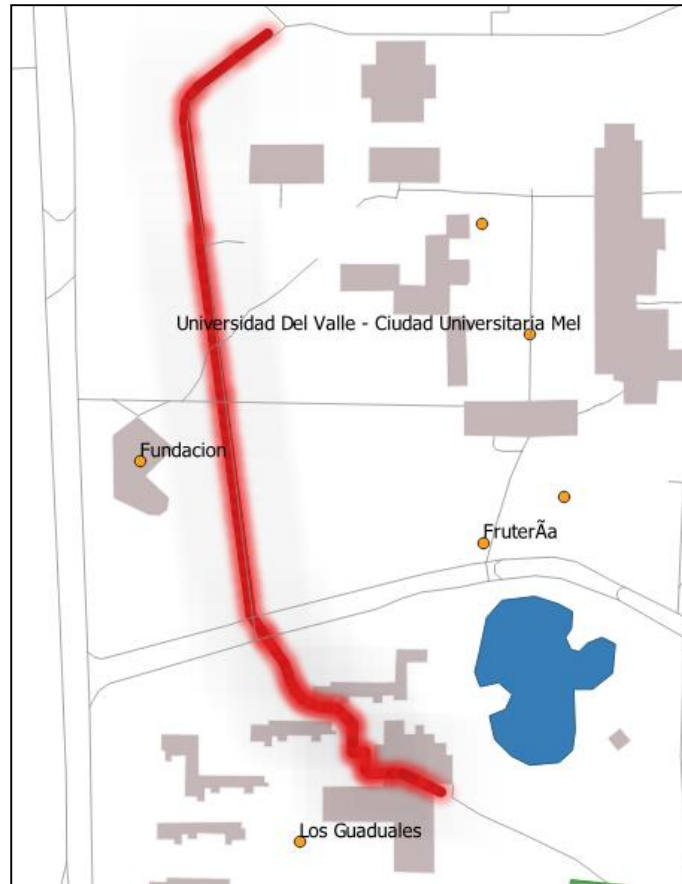
3. Calculo de la ruta más corta entre punto inicial dado por coordenadas (lat,lon) y un nodo escogido aleatoriamente, cuya trayectoria se demarca con una linea de color rojo.



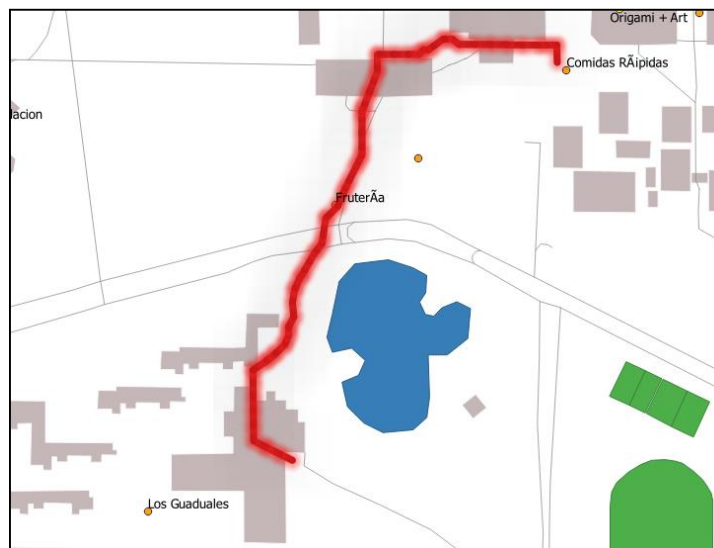
4. Cálculo entre un par de nodos dado por coordenadas (lat,lon) escogidos aleatoriamente.



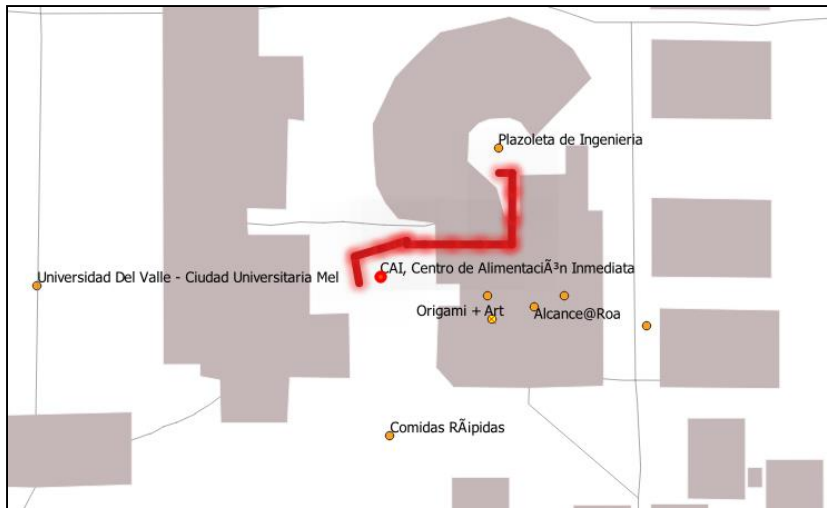
5. Cálculo ruta más corta entre punto inicial dado por coordenadas (X,Y) y punto final dado por coordenadas (lat,lon). Nota: x,y en 3115, escogidos aleatoriamente.



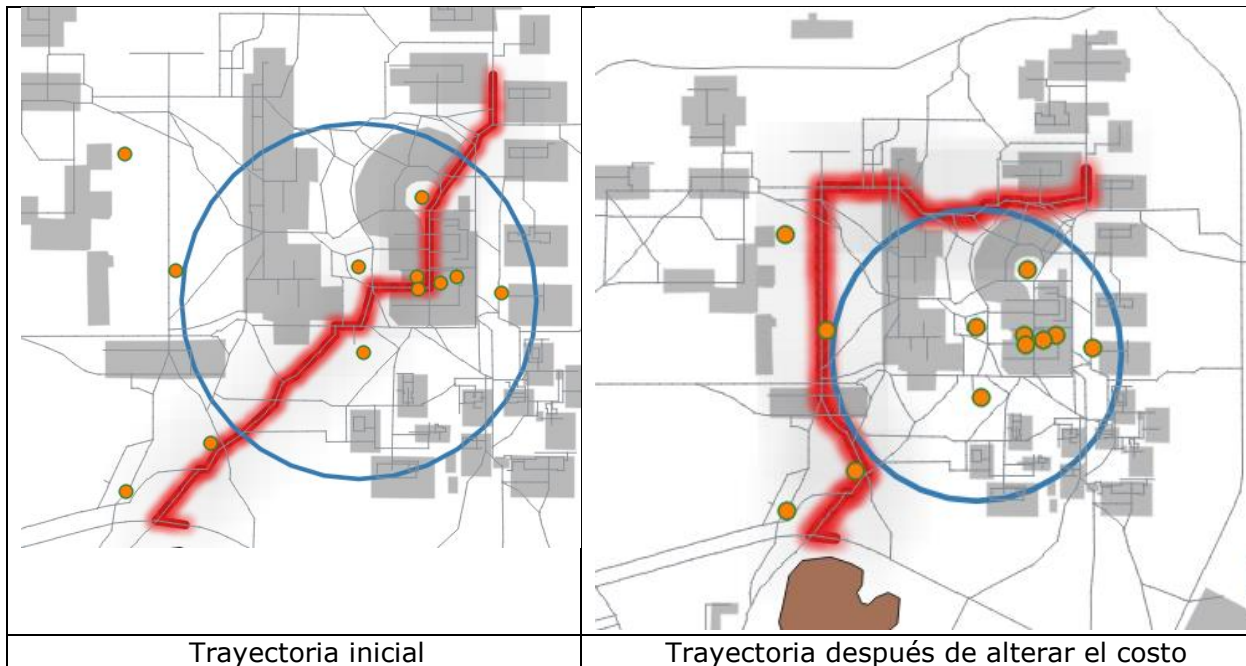
6. Cálculo entre un punto inicial dado por coordenadas (lat,lon) y el sitio de interés conocido como "Comidas R  pidas".



7. Cálculo de la Ruta más corta entre los sitios de interés "CAI" y la "plazoleta de Ingeniería".



- 8a) Calcular la ruta entre el punto $-76.53284 \ 3.37603$ y el punto $-76.53219, 3.37756$. 8b) Incrementar en un 10% el costo de los segmentos de red peatonal que se encuentran a un radio de 120 metros del punto $(-76.53289, 3.37491)$. 8c) volver a calcular la ruta del punto (a). (d) Reversar los cambios de costo.



Practica 2

Como resultado de la práctica de funciones y triggers se tiene lo siguiente:

1. Construir una función para verificar si una palabra es un palíndromo. Para ello se ingresó como ejemplo dos parámetros de prueba: `SELECT ESPalindromo('Anita lava la tina')`, `SELECT ESPalindromo('raza')`

espalindromo	
text	
1	La palabra: Anita lava la tina ES un Palíndromo

espalindromo	
text	
1	La palabra: raza NO es un Palíndromo

2. Construir una función que permita convertir coordenadas DMS a DEG. Para ello se ingresaron los valores de grados minutos y segundos así: `SELECT convertir_DMSaDEG (50,50,50)` y otra sentencia que estuviera fuera de los parámetros establecidos: `SELECT convertir_DMSaDEG (100,70,70)`.

convertir_dmsadeg	
double precision	
1	50.8472222222222

NOTICE: Verifique los parámetros ingresados

ERROR: control reached end of function without RETURN

CONTEXT: PL/pgSQL function convertir_dmsadeg(double precision,double precision,double precision)
SQL state: 2F005

3. Crear un Trigger que permita, a partir de un nuevo punto de coordenadas x,y, calcular todos los campos de la tabla referentes a: id, nombre y distancia al edificio más cercano, id, y distancia al sitio más cercano, id, y distancia al sitio más lejano, azimut , punto medio y la distancia entre los dos puntos

Data Output															Messages	Query History	Explain	Notifications
gid	x	y	edificercano_id	edificercano_nombre	edificercano_dist	sitiocercano_id	sitiocercano_dist	sitiolejana_id	sitiolejana_dist	azimut	pm_x	pm_y	distancia	the_geom				
[PK]	integer	double	double	character varying (15)	character varying (15)	double precision	double precision	double precision	double precision	double precision	double precision	double precision	double precision	geometry				

Data Output Messages Query History Explain Notifications									
gid	x	y	edificercano_id	edificercano_nombre	edificercano_dist	sitiocercano_id	sitiocercano_dist		
[PK] integer	double precision	double precision	character varying (150)	character varying (150)	double precision	character varying (150)	double precision		
1	3	-76.654	210453108	Edificio Ágora	0.0305463114481816	687147994	3415.33185107083		

CONCLUSIONES

- Postgis posee una gran variedad de herramientas de geoprocementamiento y un repositorio con la explicación de cada una, las cuales nos permiten acceder a una gran cantidad de recursos para operar capas geográficas desde query.
- PgRouting es una extensión del Sistema Gestor de Bases de Datos PostgreSQL, que permite emplear funcionalidades específicas de enrutamiento geoespacial y otras funcionalidades de análisis de redes como encontrar la ruta más corta entre dos puntos. Esta extensión relaciona el parámetro de costo que se puede alterar o modificar la ruta de marcada inicialmente. Se

requiere un buen procesador para calcular la maya de la red vial dada la cantidad de nodos o vértices que genera.

- El algoritmo de Dijkstra permite la determinación del camino más corto, dado un vértice origen, proporciona información a los nodos sobre el estado de la red, logrando tomar decisiones de la ruta, a partir de parámetros como la capacidad y el retardo del enlace.
- Un trigger o disparador es un objeto muy útil para almacenarlo en la base de datos y asociarlo a cualquier tabla para tener un control ante cualquier evento de inserción, actualización o eliminación de registros en la tabla relacionada. Otra ventaja es que no requiere que el usuario lo ejecute, este se ejecutará automáticamente por una función previamente programada. El uso de disparadores de manera incorrecta ó inefectiva puede afectar significativamente al rendimiento de nuestra base de datos.

RECURSOS

- Anexo 1. Taller 1_PgRputing.sql"
- Anexo 2. Taller 2_ triggers.sql"