APLICACIÓN DE LA POTENCIALIDAD DE INTEGRACIÓN DE UN DBMS CON PYTHON Y UNA HERRAMIENTA SIG PARA LA SELECCIÓN DE LA MUESTRA EN EL ESTUDIO DE ENCUESTAS ORIGEN Y DESTINO DOMICILIARIA

Autor:

WILLIAN PIPICANO CHICANGANA

ANEXO

Universidad Distrital Francisco José de Caldas

Facultad de Ingeniería

Especialización en Sistema de Información Geográfica

2021

CONTENIDO

1 Descripción 3

1.1 Archivo secciones 3

1.2 Archivo predios 4

1.3 Módulos Python 5

1.3.1 Crear BD 5

1.3.2 Preparar BD 6

1.3.3 Asociar 7

1.3.4 Sortear 9

1 Descripción

Este documento contiene las indicaciones principales para la aplicación del código desarrollado y los insumos necesarios para el procedimiento.

Específicamente se requiere:

1. Un archivo geográfico tipo *shape* (*shp*) compuesto por polígonos (geometrías) que representan las zonas geográficas en las que sea ha divido un área poblacional específica. Como las *secciones* censales DANE en Colombia. El conjunto de archivos debe nombrarse *secciones.shp*.
2. Un archivo geográfico tipo *shape* compuesto por polígonos (geometría) que representan los *predios* o unidades de vivienda del área geográfica representan las secciones, es decir topológicamente debe estar contenido en el área total de las secciones. El archivo debe nombrarse *predios.shp*.
3. Los módulos Python que realizan el procesamiento de los archivos anteriores. Le código que compone cada módulo se incluyen en este documento. Son cuatro módulos: *crearbd.py, prepararbd.py, asociar.py, sortear.py*. El documento principal explica el uso y orden de ejecución de cada módulo.

1.1 Archivo secciones

Archivo geográfico con las siguientes características:

Nombre: *secciones.shp*

Almacenamiento: ESRI shapefile

Codificación: UTF-8

Geometría: Polígono (Multipolígono)

SRC: EPSG:3116 - MAGNA-SIRGAS / Colombia Bogota zone – Proyectado (Puede ser otro, pero debe ser mismo en los dos archivos de trabajo)

Atributos: Mínimo el identificador de cada objeto.

1.2 Archivo predios

Archivo geográfico con las siguientes características:

Nombre: *predios.shp*

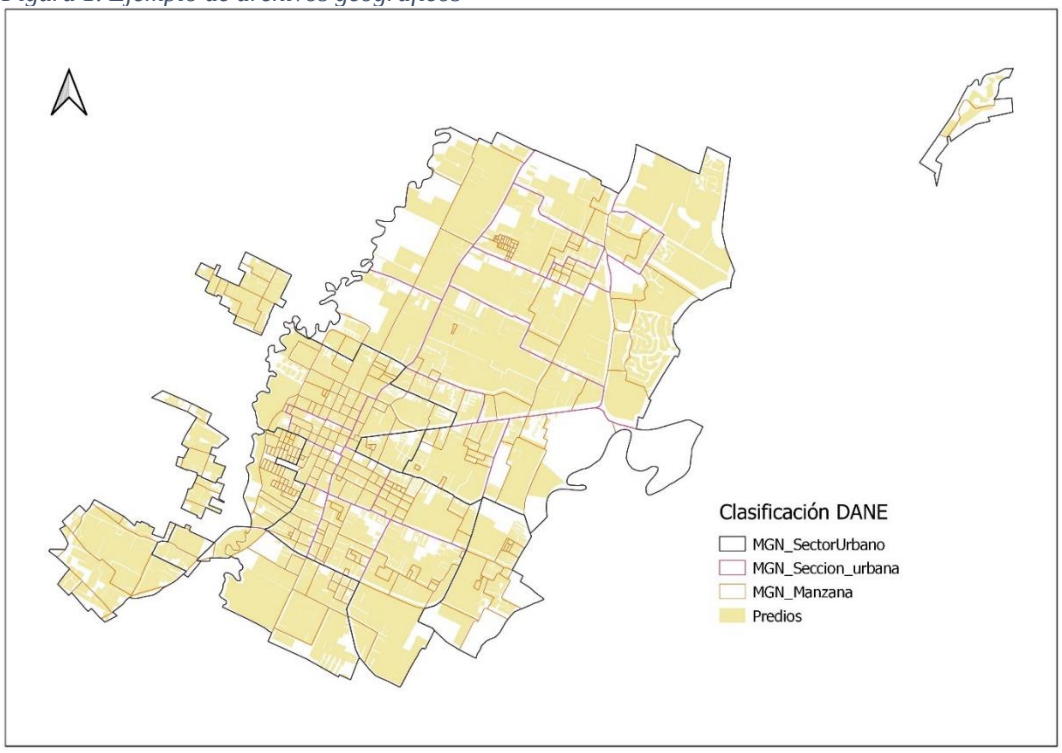
Almacenamiento: ESRI shapefile

Codificación: UTF-8

Geometría: Polígono (Multipolígono)

SRC: EPSG:3116 - MAGNA-SIRGAS / Colombia Bogota zone – Proyectado (Puede ser otro, pero debe ser mismo en los dos archivos de trabajo)

Figura 1. Ejemplo de archivos geográficos



Fuente: Elaboración propia con Datos DANE (2018) y Movilidad Sostenible Ltda (2018).

1.3 Módulos Python

Los módulos deben ejecutarse desde la terminal de comandos de Windows (para otros sistemas operativos consulte la documentación correspondiente) y el orden; 1° *crearba.py*, 2° *preparabd.py*, 3° *asociar.py* y 4° *sortear.py*. A continuación, el código de cada módulo:

1.3.1 Crear BD

```
#-------------------------------------------------------------------------------
# Name:        crearbd.py
# Purpose:     crear la base de datos
#
# Author:      William Pipicano
#
# Created:     18/03/2019
# Copyright:   (c) wpc 2021
# Licence:     libre. Dar créditos.
#-------------------------------------------------------------------------------

# Importar librería psycopg2

import psycopg2

# Establecer conexión

conexion=psycopg2.connect(
    host="localhost",
    user="postgres",
    password="postgres"
    )

conexion.autocommit= True

# Crear cursor

cur=conexion.cursor()

# Crear BD

try:
    cur.execute("CREATE DATABASE smeodh")
    print("BD creada satisfactoriamente.")

except:
    print("La BD smeodh ya existe")

finally:
    # Cerrar conexion
    conexion.close()
```

1.3.2 Preparar BD

```
#-------------------------------------------------------------------------------
# Name:        prepararbd.py
# Purpose:     hacer ajustes en la bd necesarios para el procesamiento
#
# Author:      William Pipicano
#
# Created:     18/03/2019
# Copyright:   (c) wpc 2021
# Licence:     libre. Dar créditos.
#-------------------------------------------------------------------------------

# ESTE MODULO DEBE EJECUTARSE DESDE LA TERMINAL DE COMANDOS DE WINDOWS

# Importar librería psycopg2

import psycopg2
import os

# Crear conexión con la base de datos

conexion=psycopg2.connect(
    database="smeodh",
    host="localhost",
    user="postgres",
    password="postgres"
    )

# Crear cursor

cur=conexion.cursor()

# Dotar la BD de la funcionalidad espacial

try:

    cur.execute("CREATE EXTENSION postgis")

    print("La funcionalidad espacial de la BD ha sido agregada
satisfactoriamente.")

except:

    print("La extensión PostGIS ya existe.")


# Agregar tablas a la BDE desde los shapes secciones y predios.

# Si las instrucciones siguientes retornan error, debe configurar la ruta
# correspondiente en la variable de entorno PATH. La ruta de ubicación del
# archivo shape por transformar debe corresponder.
```

```
# Agregar tabla secciones

try:

    os.system("shp2pgsql -s 3116 -I \
              C:\\proyecto\\shp\\secciones secciones > secciones.sql")

    os.system("psql -h localhost -d smeodh -U postgres -f secciones.sql")

    print("Tabla secciones agregada exitosamente.")

except:

    print("La tabla secciones ya existe o se presentó un error. Revisar.")

# Agregar tabla predios

try:

    os.system("shp2pgsql -s 3116 -I \
              C:\\proyecto\\shp\\predios predios > predios.sql")

    os.system("psql -h localhost -d smeodh -U postgres -f predios.sql")

    print("Tabla predios agregada existosamente")

except:

    print("La tabla predios ya existe o se presentó un error. Revisar.")

finally:

    # Hacer que los cambios en la base de datos sean persistentes y cerrar
    conexión

    conexion.commit()
    conexion.close()
```

1.3.3 Asociar

```
#-------------------------------------------------------------------------------
# Name:        asociar.py
# Purpose:     Asociar predios a secciones
#
# Author:      William Pipicano
#
# Created:     18/03/2019
# Copyright:   (c) wpc 2021
# Licence:     libre
#-------------------------------------------------------------------------------
```

```
'''
Asociación de los predios y las secciones asignando el gid (idenficador) de las
secciones al conjunto de predios contenidos en cada sección de forma
correspondiente

'''

# Importar librería psycopg2

import psycopg2

# Crear conexión con la base de datos

conexion=psycopg2.connect(
    database="smeodh",
    host="localhost",
    user="postgres",
    password="postgres"
    )

# Crear cursor

cur=conexion.cursor()

# Ajustar tabla predios, se agrega los campos que contendrán ids (gid_sec de las
# secciones y la indenficación (ecuestar) del prediosseleccionado para encuesta

cur.execute("ALTER TABLE predios \
    ADD gid_seccion INTEGER REFERENCES secciones(gid), \
    ADD seleccion INTEGER")

# Obtener gid: identificador de secciones

cur.execute("SELECT gid FROM secciones")

# Guardar la consulta en una variable

gids_sec=cur.fetchall()

# Declaramos una variable tipo lista para almacenar el resultado del proceso

resultado=[]

# Consulta de tabla predios y tabla secciones: se realiza iteración por cada
# sección al tiempo que se crea una lista que asigne el gid de la sección al
# conjunto de predios contenido en la sección correspondiente

for i in range(len(gids_sec)):
    sql="SELECT pr.gid, pr.gid_seccion FROM predios AS pr, secciones AS s \
    WHERE st_contains(s.geom,pr.geom) AND s.gid=%s;"
    val=gids_sec[i]

    cur.execute(sql,val)

    resultado_parcial=cur.fetchall()
```

```
    for j in range(len(resultado_parcial)):
        temp=list(resultado_parcial[j])
        temp[1]=val[0]
        elemento_j=tuple(temp)
        resultado.append(elemento_j)

# Reporte de elementos procesados
print(str(len(resultado)) + " predios asociados")

# Crear tabla (pred_sec) de apoyo para el proceso, a partir del resultado (lista)
# que contendrá los gid de los predios y la sección a la que pertence cada uno

cur.execute("CREATE TABLE pred_sec (gid_predio INTEGER PRIMARY KEY, \
gid_sec INTEGER)")

# Insertar valores a tabla (pred_sec) creada en el paso anterior

sql="INSERT INTO pred_sec (gid_predio,gid_sec) VALUES (%s,%s)"
val=resultado
cur.executemany(sql,val)

# Llevar los valores de sección de pred_sec a la tabla predios

cur.execute("UPDATE predios SET gid_seccion=gid_sec FROM pred_sec \
WHERE predios.gid=pred_sec.gid_predio")

# Confirmar proceso

print("Proceso realizado con éxito...")

# Hacer que los cambios en la base de datos sean persistentes y cerrar conexion
conexion.commit()
conexion.close()
```

1.3.4 Sortear

```
#-------------------------------------------------------------------------------
# Name:        sortear.py
# Purpose:     Selección aleatoria de unidades por encuestar (predios)
#
# Author:      William Pipicano
#
# Created:     18/03/2019
# Copyright:   (c) wpc 2021
# Licence:     libre
#-------------------------------------------------------------------------------

# Importar librerias

import psycopg2 # conexión
import random # procesos aleatorios
```

```
# Crear conexión con la base de datos

conexion=psycopg2.connect(
    database="smeodh",
    host="localhost",
    user="postgres",
    password="postgres"
    )

# Crear cursor

cur=conexion.cursor()

# Obtener información de las unidades (predios) para proceso de selección

cur.execute("SELECT gid, gid_seccion FROM predios")

predios_base=cur.fetchall()

# Ordenar lista por campo gid_seccion

predios_orden=sorted(predios_base, key=lambda seccion:seccion[1])

'''
Crear grupos de unidades (predios) por zona (sección), se declara primero el
contenedor de la población para la seleccion muestral y punto de referencia
para procesamiento
'''
conjunto_ppal=[]
referencia=[predios_orden[0]]
k=0

n=len(predios_orden)

# Crear lista compuesta con sublista de tuplas por zona (sección)

for i in range(n-1):

    if predios_orden[i+1][1]==referencia[k][1]:
        referencia.append(predios_orden[i+1])
        k+=1
    else:
        conjunto_ppal.append(referencia)
        referencia=[predios_orden[i+1]]
        k=0

conjunto_ppal.append(referencia)

# Obtener cantidad de muestra por zona (sección)

sql="SELECT gid_seccion, COUNT(gid), ROUND(COUNT(gid)*%s,0) as muestra \
FROM predios GROUP BY gid_seccion ORDER BY gid_seccion"

# Solicitar el tamaño de muestra n
```

```
np=float(input("Tamaño de muestra n [%] = "))
val_n=[]
val_n.append(np/100)
val=tuple(val_n)

cur.execute(sql,val)
distribucion_zona=cur.fetchall()

# Selección de la muestra

muestra_zona=[]

for i in range(len(conjunto_ppal)):
    muestra_i=random.sample(conjunto_ppal[i],int(distribucion_zona[i][2]))
    for j in range(len(muestra_i)):
        muestra_zona.append(muestra_i[j])

# Total muestra
print(len(muestra_zona), 'viviendas')

# Antes limpiar campo seleccion

cur.execute("UPDATE predios SET seleccion=null")

# Marcar las unidades seleccionadas (predios) en la tabla predios

for k in range(len(muestra_zona)):
    sql_marca="UPDATE predios SET seleccion=1 WHERE gid=%s"
    valor=(muestra_zona[k][0],)
    cur.execute(sql_marca,valor)

# Confirmación del proceso
print("La selección aleatoria se ha realizado satisfactoriamente. Puede \
visualizar el resultado en QGIS o la aplicación de preferencia")

# Hacer que los cambios en la base de datos sean persistentes y cerrar conexion

conexion.commit()
conexion.close()
```