PDET Municipality Boundaries Dataset Integration

Tercera entrega

Eliana Katherine Cepeda
Juan Diego Gonzalez
Mateo Moreno
Noviembre 9, 2025

1. Resumen Ejecutivo

Objetivo de la entrega. Documentar el estado de avance de la fase Google (Open Buildings) del pipeline de integración de edificaciones para municipios PDET, dejando lista la base de datos NoSQL, los controles de calidad y las plantillas de métricas de eficiencia.

Fuentes procesadas en esta fase.

- Google Open Buildings (GOB). (Microsoft se incluirá en la siguiente iteración.)

Estado de avance.

- Estructura NoSQL (MongoDB) y esquema propuesto: **listo** (en esta entrega).
- Índices espaciales 2dsphere: **listo** (definición y convenciones).
- Pipeline de carga (lectura por chunks, filtro PDET): **listo**
- Métricas de rendimiento (plantilla y definiciones): **listo**.
- EDA inicial (plantilla): **listo**.

Hallazgos preliminares:

- Estructura de campos clave detectados: geometry, area_in_meters, confidence.
- Se prioriza CRS **EPSG:4326** con geometrías válidas Polygon/MultiPolygon.

2. Datos y Fuentes (Google y Microsoft)

Como se explica en el diseño para este entregable se usan las fuentes de Google y de Microsoft para recopilar los datos de las geometrías de los edificios que hay en Colombia para filtrarlos posteriormente.

2.2. Estructuras de archivo y particionamiento

- Esquema de nombres: por tiles/país/área; verificar patrón de archivos del dump de Google usado en la corrida.
- Formatos esperados: CSV (posible compresión ZIP) o GeoJSON por tile.
- Si CSV: geometry usualmente en WKT; se estandariza a GeoJSON.

3. Arquitectura y Esquema NoSQL

3.1. Resumen de la arquitectura

- Motor: MongoDB (NoSQL geoespacial).
- Colecciones propuestas: municipios, edificaciones (campo fuente: google/microsoft).

3.2. Esquema de colección de edificaciones

Campos mínimos:

- fuente: string (google | microsoft).
- geometria: GeoJSON (Polygon | MultiPolygon), CRS EPSG:4326.
- area_m2: number.
- confianza: number (si aplica; para Google mapea confidence).

Plantilla de mapeo por fuente:

- Google → esquema: geometry→geometria, area_in_meters→area_m2, confidence→confianza.
- Microsoft → esquema: geometry→geometria.

3.3. Indexación espacial

- Índice 2dsphere sobre geometria.
- Índices secundarios: { fuente: google/ microsoft}.

3.4. Controles de integridad

- Validación de tipo geométrico y is_valid (buffer(0) si se requiere fix).
- Normalización de CRS a EPSG:4326.
- Reglas para area_m2 > 0 y límites superiores por percentil.

4. Pipeline de Carga e Integración

4.1. Flujo de procesamiento

1. Ingesta por chunks desde CSV o GeoJSON.
2. Parseo de geometrías WKT → GeoJSON (Shapely/GeoPandas) y unificación de CRS.
3. Filtro espacial por PDET (intersección con unary_union o *spatial join* a límites municipales DANE/MGN).
4. Limpieza/normalización de atributos (area_m2, confianza, mapeo de nombres).
5. Inserción bulk en MongoDB (insert_many(ordered=False)).

4.2. Reproducibilidad

- Rutas de notebooks/scripts y variables de entorno documentadas.
- Logs de ejecución por archivo (TPA, TPS, docs/min, %PDET).
- Seeds/versions de librerías.

5. Data Loading Efficiency (Métricas de Rendimiento)

5.1. KPIs y definiciones

- TPA (tiempo por archivo) [min].
- TPS (tiempo por 100k filas) [min/100k].
- Tasa de inserción [docs/min].
- % filtrado PDET (doc. insertados / doc. leídos).
- Uso pico de RAM (si está disponible).

5.2. Resultados por lote/archivo

Fuente	Archivo/Tile	Filas leídas	Edificios PDET	TPA (min)	Docs/min	% PDET
Datos Google	*8df_buildings.csv*	22,806	0	0.01min	0	0%
Datos Google	*8e1_buildings.csv*	277,823	80,565	252.61min	319 Aprox	29% Aprox
Datos Google	*8e3_buildings.csv*	1,410,500 Aprox	196,339	1,474min	133.2	14% Aprox
Datos Microsoft	*Colombia.geojson*	458,000	94,552	9.35 min	10,113	21% Aprox

6. Cuadro comparativo de EDA

Datos Microsoft	Datos Google

{
"_id": "estadisticas_de_area",
"areaTotal": 13,163,900.416474573,
"areaPromedio": 139.1709351764978,
"areaMinima": 20.00321613058856,
"areaMaxima": 55,227.85991916092
}
	{
	"_id": "estadisticas_de_area",
	"areaTotal": 8,819,739.559,
	"areaPromedio": 88.19739559,
	"areaMinima": 2.752,
	"areaMaxima": 6,693.307
	}

7. Serie de pasos

Datos de Microsoft

Primero se descargaron los datos del Github, abajo se encontraban los datos divididos por país, así que se seleccionó Colombia, se guardó en descargar para posteriormente preparar el entorno en visual Studio Code. Donde empezamos importando las librerías necesarias para procesar los datos:

```
import pandas as pd
import geopandas as gpd
import pymongo
from pymongo import MongoClient
import json
import os
import time
from shapely.geometry import shape
import itertools
import gc

print("Librerías importadas correctamente.")
```

```
Librerías importadas correctamente.
```

Posteriormente se hicieron los ajustes para crear las conexiones en Mongo, en este caso hicimos doble conexión, una para lectura y otra local para escritura. Esto se hizo porque todo lo que hicimos en la entrega 2, estaba en un Mongo, así que se leyó ese y se trabajó localmente.

```
Conectado a BD (Lectura): 'is394508_db'
Conectado a BD (Escritura): 'is394512_db'
¡Ambas conexiones a MongoDB exitosas!
```

Después, desde el Mongo de lectura se cargaron los Municipios PDET unificando todas las geometrías PDET en una sola y se muestra la carga de los 170 municipios para luego cerrar la sesión de lectura.

```
Iniciando la carga de municipios PDET desde MongoDB (Lectura)...
Unificando todas las geometrías PDET en una sola (unary_union)...
C:\Users\Simon Esteban G\AppData\Local\Temp\ipykernel_23628\788945061.py:29:
  filtro_pdet_unificado = gdf_municipios_pdet.geometry.unary_union
¡Filtro PDET listo! (170 municipios cargados en 12.66s)
Conexión de Lectura  cerrada.
```

Se creó la colección en el Mongo local "*edificios_microsoft_pdet"* y se creó el índice espacial 2dsphere y se cargó el archivo GeoJson con los datos.

```
Preparando colección de destino 'edificios_microsoft_pdet'...
Creando índice espacial '2dsphere' (sparse=True)...
¡Índice 2dsphere creado en 0.04s!
```

Asimismo, se procedió con el ETL cin Chunks de 500. En los datos de Microsoft se hizo una procedimiento distinto que fue el cálculo del área ya que para compararlo con los datos de Google (Que si tenía el área), era necesario este cálculo, fue un poco complicado porque el Json solo tenia coordenadas. Para calcular el área de cada edificio, fue necesario realizar un paso de transformación clave durante el proceso de carga. Las geometrías originales del archivo de Microsoft se encuentran en un sistema de coordenadas *(EPSG:4326)* cuyas unidades son grados de latitud y longitud. Si calculamos el área directamente con estas unidades, el resultado no estaría en metros. Por lo tanto, para obtener un área útil en metros cuadrados (m²), el script "reproyecta" temporalmente la geometría de cada edificio a un sistema de coordenadas oficial para Colombia que sí utiliza metros *(EPSG:3116)*. Una vez la geometría estuvo transformada a este sistema, se usó la función .area de Geopandas para calcular la superficie real. Este valor en metros cuadrados es el que finalmente se almacenó en el campo area_m2 de cada documento en nuestra base de datos MongoDB.

```
...
Tiempo total: 567.00 segundos
Total de líneas (edificios) procesadas del archivo: 458000
Total de edificios PDET cargados en tu BD: 94552
Total de edificios con geometrías inválidas (omitidos): 0
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Datos Google

Se hizo el mismo proceso anterior, solo que no se hizo la doble conexión con MongoDB, sino todo local, además de que se procesaron diversos archivos (9 Archivos) y no se hizo el cálculo de las áreas, porque ya venía en los datos.

Finalmente, se pasó el JSON de los datos de Microsoft con los Edificios PDET y se cargaron al mismo MongoDB para tener todo en la misma base de datos

8. Conclusiones

- No podemos cargar todos los datos como nos hubiera gustado, por limitaciones de PC y mala gestión pero nos deja de enseñanza para planificar mejor la carga de los datos, no éramos conscientes de este problema.

- La lección principal de esta entrega es que la estrategia de "Filtrar-Luego-Cargar" (ETL) en Python no es escalable para este volumen de datos. Para la Entrega 4, se planea una arquitectura de "Cargar-Luego-Filtrar" (ELT):

    - **Cargar Todo:** Cargar los 6 millones de edificios de Microsoft directamente a MongoDB en una colección temporal, sin ningún filtro.

    - Filtrar en la BD: Usar el poder de los índices 2dsphere de MongoDB (con el operador $geoIntersects) para que sea la base de datos, y no Python, quien realice el filtro.