Carga del modelo documental en MongoDB

En este notebook se realiza la inserción del modelo documental generado previamente en una base de datos MongoDB local.

Antes de insertar los documentos se elimina la colección existente para evitar duplicados y garantizar que cada ejecución del proceso parta de un estado limpio, equivalente al uso de DROP TABLE en bases de datos relacionales.

Posteriormente se cargan los documentos en formato JSON Lines y se insertan en la colección `locales`.

In [ ]:
```
# Si no está instalado
# !pip install pymongo
```

Conexión a MongoDB

Se establece conexión con una instancia local de MongoDB ejecutándose en el puerto por defecto (27017).

In [1]:
```
from pymongo import MongoClient
import json
from pathlib import Path
```

In [2]:
```
client = MongoClient("mongodb://localhost:27017/")

print("Conectado a MongoDB")
print(client.list_database_names())
```

```
Conectado a MongoDB
['admin', 'config', 'local', 'prueba']
```

Creación de base de datos y colección

MongoDB crea automáticamente la base de datos y la colección cuando se insertan documentos por primera vez. No obstante, primero se elimina la colección existente para evitar duplicidades entre ejecuciones.

In [ ]:
```
db = client["actividad_comercial_madrid"]

# Si existe la colección la borramos
if "locales" in db.list_collection_names():
    db.locales.drop()
    print("Colección previa eliminada")

coleccion = db["locales"]
```

Problema detectado: pérdida del tipo fecha en MongoDB

Durante la primera carga de datos se observó que los campos temporales (fechas de licencias y terrazas) no se almacenaban en MongoDB como tipo `Date`, sino como cadenas de texto.

Esto ocurre porque el formato JSON no dispone de un tipo nativo de fecha. Al exportar los documentos a JSON Lines, los valores `datetime` de Python se serializan automáticamente a texto.

Ejemplo:

datetime (Python) → "2022-05-14 00:00:00" (JSON) → String en MongoDB

Como consecuencia, las consultas que filtran por rangos temporales no funcionan correctamente usando operadores como `$gte` o `$lt`, ya que MongoDB no puede comparar fechas reales con strings.

Para solucionar este problema, durante la carga se implementa una transformación previa que detecta automáticamente valores con formato de fecha y los convierte nuevamente a objetos `datetime` de Python antes de insertarlos en MongoDB.

De este modo, MongoDB almacena los campos como `ISODate`, permitiendo realizar consultas temporales eficientes sin conversiones adicionales en los pipelines.

In [5]:
```
from datetime import datetime
import re

# Detecta formatos de fecha comunes del dataset Madrid
regex_fecha = re.compile(
```

```
    r"^\d{4}-\d{2}-\d{2}|\d{2}/\d{2}/\d{4}|\d{4}-\d{2}-\d{2}T\d{2}:\d{2}"
)

def convertir_valor(valor):
    """Convierte strings con formato fecha a datetime"""
    if isinstance(valor, str) and regex_fecha.match(valor):
        for fmt in ("%Y-%m-%d %H:%M:%S",
                    "%Y-%m-%d",
                    "%d/%m/%Y",
                    "%Y-%m-%dT%H:%M:%S"):
            try:
                return datetime.strptime(valor[:19], fmt)
            except:
                pass
    return valor


def convertir_documento(doc):
    """Recorre recursivamente el documento convirtiendo fechas"""
    if isinstance(doc, dict):
        return {k: convertir_documento(v) for k, v in doc.items()}
    elif isinstance(doc, list):
        return [convertir_documento(v) for v in doc]
    else:
        return convertir_valor(doc)
```

Lectura del archivo JSONL

Se carga el archivo generado en la fase de modelado documental. Cada línea representa un documento independiente.

In [6]:
```
ruta = Path("../../data/output/locales.jsonl")

documentos = []

with open(ruta, "r", encoding="utf-8") as f:
    for linea in f:
        doc = json.loads(linea)
        doc = convertir_documento(doc)
        documentos.append(doc)
```

```
print("Documentos cargados:", len(documentos))
```

```
Documentos cargados: 151162
```

Inserción de documentos

Los documentos se insertan mediante `insert_many`, lo que permite una carga masiva eficiente.

In [7]:
```
resultado = coleccion.insert_many(documentos)

print("Insertados:", len(resultado.inserted_ids))
```

```
Insertados: 151162
```

Verificación de la carga

Se comprueba que los documentos han sido almacenados correctamente en la colección.

In [8]:
```
coleccion.count_documents({})
```

Out[8]: 151162

In [9]:
```
coleccion.find_one()
```

Out[9]:

```
{'_id': ObjectId('6997606d55b23c51750f7f59'),
 'id_local': 20000596,
 'desc_distrito_local': 'ARGANZUELA',
 'desc_barrio_local': 'CHOPERA',
 'desc_tipo_acceso_local': 'Puerta Calle',
 'desc_situacion_local': 'Abierto',
 'clase_vial_edificio': 'CALLE',
 'desc_vial_edificio': 'TOMAS BORRAS',
 'rotulo': 'LA CAÑA',
 'cod_postal': 28045,
 'coordenada_x_local': 440788.6,
 'coordenada_y_local': 4471857.5,
 'hora_apertura1': '09:00',
 'hora_cierre1': None,
 'licencias': [{'ref_licencia': '500/2016/12956',
   'desc_tipo_licencia': 'Declaración Responsable',
   'desc_tipo_situacion_licencia': 'En tramitación',
   'Fecha_Dec_Lic': datetime.datetime(2016, 9, 29, 0, 0)},
  {'ref_licencia': '350/2022/03432',
   'desc_tipo_licencia': 'Transmisión de licencia Urbanística',
   'desc_tipo_situacion_licencia': 'Transmisión de Licencia Concedida',
   'Fecha_Dec_Lic': datetime.datetime(1900, 1, 1, 0, 0)}],
 'terrazas': [],
 'actividad_economica': [{'desc_seccion': 'COMERCIO AL POR MAYOR Y AL POR MENOR; REPARACIÓN DE VEHÍCULOS DE MOTOR Y MOTOCICLE
TAS',
   'desc_division': 'COMERCIO AL POR MENOR, EXCEPTO DE VEHÍCULOS DE MOTOR Y MOTOCICLETAS',
   'desc_epigrafe': 'COMERCIO AL POR MENOR DE PRENDAS DE VESTIR EN ESTABLECIMIENTOS ESPECIALIZADOS'}]}
```