Universidad Complutense de Madrid

Máster Big Data & Data Engineering

Bases de Datos NoSQL

Actividad Práctica

Profesor:
Marlon Cárdenas

Andrés Vega Zamora

13 de febrero de 2025

Índice

1. Exploración inicial de los datos

1.1. Revisión inicial

Fichero Actividad económica y Locales:
Provenientes del Censo de Actividades y Locales de la Comunidad de Madrid. Se les asocian características físicas de los locales, actividades realizadas en el mismo y sus titulares.
Se cuenta con dos ficheros:

- Fichero de locales: Cada registro corresponde a un local.
- Fichero de actividades: Cada registro corresponde a un local más un código de actividad. Un local se repite tantas veces como actividades asociadas tenga.

Fichero Licencias:
Provee información física de los locales además de información sobre las licencias a concedidas a cada local.
Fichero Terrazas:
Provee información física de las terrazas y los locales a los que pertenecen.

1.2. Revisión de campos

Se realiza una limpieza de datos en todos los ficheros para evitar datos que no brindan información útil y datos repetidos.
Fichero locales
Se eliminan columnas que tienen muy pocos datos no nulos ("hora_apertura2", "hora_cierre2"):

```
df_locales = df_locales.drop(['hora_apertura2', 'hora_cierre2'], axis = 1, )
```

Se eliminan columnas que tienen un único valor en toda la columna:

```
df_locales= df_locales.drop(['fx_datos_ini','fx_datos_fin', '
    id_clase_ndp_edificio', 'id_clase_ndp_acceso'], axis = 1);
```

Se eliminan los leading y trailing whitespaces de las columnas de texto:

```
for column in df_locales.columns:
    if df_locales[column].dtype == "object":
        df_locales[column] = df_locales[column].str.strip()
```

Se modifica el formato de ubicación geográfica a latitud y longitud:

```
from pyproj import CRS, Transformer

# 1. Define la proyeccion UTM (zona 30N para Madrid) y la proyeccion geografica (
    WGS84)
utm_crs = CRS("EPSG:25830")  # UTM Zone 30N
wgs84_crs = CRS("EPSG:4326")  # WGS84 (latitud y longitud)

```

```
# 2. Crea el transformador de coordenadas
transformer = Transformer.from_crs(utm_crs, wgs84_crs)

# 3. Funcion para convertir coordenadas UTM a latitud y longitud
def utm_to_latlon(x, y):
    lat, lon = transformer.transform(x, y)
    return lat, lon

df_locales[['latitud', 'longitud']] = df_locales.apply(lambda row: utm_to_latlon(
    row['coordenada_x_local'], row['coordenada_y_local']), axis=1, result_type='
    expand')
```

Fichero actividades

Se convierte el tipo de columnas 42 y 44 a int64, ya que todos menos uno de sus valores son numéricos:

```
df_actividades['id_division'] = pd.to_numeric(df_actividades['id_division'],
    errors = 'coerce')
df_actividades['id_epigrafe']  = pd.to_numeric(df_actividades['id_epigrafe'],
    errors = 'coerce')
df_actividades['id_division'] = df_actividades['id_division'].replace(np.nan,
    999).infer_objects(copy=False)
df_actividades['id_epigrafe'] = df_actividades['id_epigrafe'].replace(np.nan,
    999).infer_objects(copy=False)
df_actividades['id_division'] = df_actividades['id_division'].astype('Int64')
df_actividades['id_epigrafe'] = df_actividades['id_epigrafe'].astype('Int64')
```

Se nota que cuando el id_epigrafe es -1, no hay informacion del local. Se eliminan esas filas:

```
df_actividades = df_actividades[df_actividades["id_epigrafe"] != -1]
```

Fichero terrazas

Se eliminan los leading y trailing whitespaces de las columnas de texto:

```
for column in df_locales.columns:
    if df_locales[column].dtype == "object":
        df_locales[column] = df_locales[column].str.strip()
```

Fichero licencias

Se cambia el formato de la fecha a ISO Date YYYY-MM-DD:

```
df_licencias["Fecha_Dec_Lic"] = pd.to_datetime(df_licencias["Fecha_Dec_Lic"],
    format = '%d/%m/%Y')
df_licencias["Fecha_Dec_Lic"] = df_licencias["Fecha_Dec_Lic"].dt.strftime('%Y-%m
    -%d')
```

1.3. Relación entre ficheros

Luego de observar los datos pertenecientes a cada fichero se nota que todos los ficheros tienen muchas columnas que se repiten con respecto al fichero de locales, y estas columnas dan información física

sobre los locales en sí, no de las terrazas, licencias o actividades. De manera que se procede a eliminar las columnas repetidas.

```
#Se eliminan columnas repetidas entre fichero locales y actividades, del fichero de actividades

columns_locales_actividades = set(df_locales.columns) & set(df_actividades.columns)
columns_locales_actividades.remove('id_local')
df_actividades = df_actividades.drop(columns_locales_actividades, axis = 1)
```

Se sigue el mismo proceso para las columnas entre locales-terrazas y locales-licencias. De manera cada uno de estos ficheros ahora solo brinda información sobre su propio ámbito, y la relación con el fichero de locales se establece por medio de la columna "id_local" que mantienen todos los ficheros.

2. Modelado de datos

2.1. Diseño del modelo de datos

Se decide utilizar un modelo desnormalizado/embebido en los locales para ingresar los datos en MongoDB. Se prioriza la eficiencia en la consulta de datos sobre la eficiencia en la actualización de datos.

Se establece la conexión con Mongo:

```
client = MongoClient("mongodb://localhost:27017/")
db = client["locales_madrid_mongo"]
locales_collection = db["locales_madrid"]
```

Se convierten los dataframes a listas de diccionarios:

```
datos_locales = df_locales.to_dict(orient = "records")
datos_actividades = df_actividades.to_dict(orient = "records")
datos_licencias = df_licencias.to_dict(orient = "records")
datos_terrazas = df_terrazas.to_dict(orient = "records")
```

Construcción del modelo embebido:

```
datos_embebidos = {}

for actividad in datos_actividades:
    id_local = actividad["id_local"]
    del actividad["id_local"] # Se elimina el id_local del diccionario de actividad porque ya la actividad va a estar embebida dentro del local
    datos_embebidos.setdefault(id_local, {"actividades" : [], "licencias" : [], "terraza": []}) #Vacio si no existen los datos
    datos_embebidos[id_local]["actividades"].append(actividad)

```

```
for licencia in datos_licencias:
    id_local = licencia["id_local"]
    del licencia["id_local"]
    datos_embebidos.setdefault(id_local, {"actividades" : [], "licencias" : [], "terraza": []}) #Vacio si no existen los datos
    datos_embebidos[id_local]["licencias"].append(licencia)

for terraza in datos_terrazas:
    id_local = terraza["id_local"]
    del terraza["id_local"]
    datos_embebidos.setdefault(id_local, {"actividades" : [], "licencias" : [], "terraza": []}) #Vacio si no existen los datos
    datos_embebidos[id_local]["terraza"].append(terraza)
```

Inserción de datos a Mongo:

```
for local in datos_locales:
    id_local = local["id_local"]
    info_embebida = datos_embebidos.get(id_local)
    if info_embebida:
        local.update(info_embebida)
    locales_collection.insert_one(local)
client.close()
```

La colección resultante tiene la siguiente estructura:

```
{
  "_id": {"$oid": "67a7ad7ab7911aea08fe75b7"},
  "id_local": 40002093,
  "id_distrito_local": 4,
  "desc_distrito_local": "SALAMANCA",
  "id_barrio_local": 406,
  "desc_barrio_local": "CASTELLANA",
    ...
  "actividades": [
    {
      "id_seccion": "I",
      "desc_seccion": "HOSTELERÍA",
      "id_division": 56,
      "desc_division": "SERVICIOS DE COMIDAS Y BEBIDAS",
      "id_epigrafe": 561005,
      "desc_epigrafe": "BAR CON COCINA"
    }
  ],
  "licencias": [
```

```
    {
      "ref_licencia": "350/2023/36134",
      "id_tipo_licencia": 4,
      "desc_tipo_licencia": "Transmisión de licencia Urbanística",
      "id_tipo_situacion_licencia": 5,
      "desc_tipo_situacion_licencia": "Transmisión de Licencia Concedida",
      "Fecha_Dec_Lic": "1900-01-01"
    }
  ],
  "terraza": [{...}]
}
```


2.2. Validación del modelo mediante consultas

Se ejecutan las siguientes consultas para evaluar la eficiencia y funcionalidad del modelo. Se adjuntan **recortes** de los resultados de las consultas.

a - Total de locales y terrazas por distrito y barrio

```
[
  {
    $group: { //Agrupa por barrio y distrito
      _id: {
        barrio: "$desc_barrio_local",
        distrito: "$desc_distrito_local"
      },
      totalLocales: { $sum: 1 },
      totalTerrazas: { //Verifica si el local tiene terraza
        $sum: {
          $cond: {
            if: {
              $and: [
                { $not: { $eq: ["$terraza", []] } },
                { $gt: [{ $size: { $ifNull: ["$terraza", []] } }, 0] }
              ]
            },
            then: 1,
            else: 0
          }
        }
      }
    }
  },
    {
    $project: { //Selecciona campos a mostrar
```

```
      _id: 0, // Excluye el campo _id
      distrito: "$_id.distrito",
      barrio: "$_id.barrio",
      totalLocales: 1,
      totalTerrazas: 1
    }
  },
  {
    $sort: { // Ordena por distrito y barrio
      distrito: 1,
      barrio: 1
    }
  }
]
```

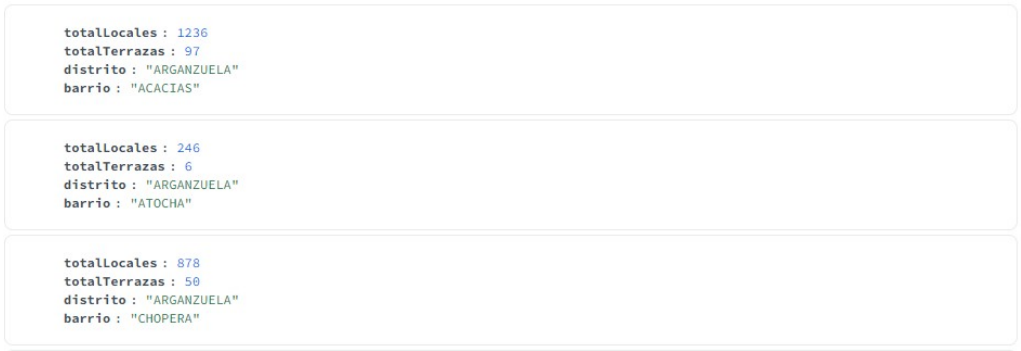

Figura 1: Locales y terrazas por distrito y barrio

b - Tipos de licencias y cantidad de licencias por tipo

```
[
  {
    $unwind: "$licencias" // Desglosa el array de licencias
  },
  {
    $group: {
      _id: "$licencias.desc_tipo_licencia", // Agrupa por tipo de licencia
      cantidad: { $sum: 1 } // Cuenta las licencias de cada tipo
    }
  },
  {
    $project: { //Selecciona los campos a mostrar
      _id: 0, // Excluye el campo _id
      tipo_licencia: "$_id",
      cantidad: 1
    }
  },
```

```
  {
    $sort: { // Ordena por cantidad de licencias de mayor a menor
      cantidad: -1
    }
  }
]
```

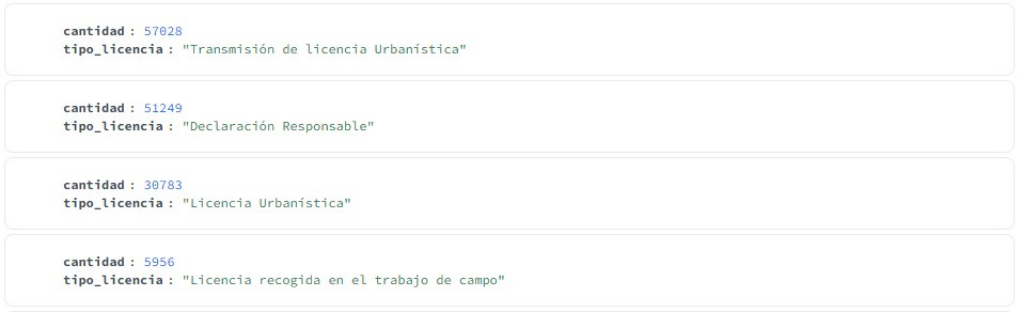

Figura 2: Tipos de licencias y cantidad

c - Listado de locales y terrazas con licencias "En tramitación"

```
[
  {
    $unwind: "$licencias" // Desglosa el array de licencias
  },
  {
    $match: { // Verifica que la licencia este "En tramitacion"
      $expr: {
        $regexMatch: {
          input:
            "$licencias.desc_tipo_situacion_licencia",
          regex: "^En tramitacion$",
          options: "i"
        }
      }
    }
  },
  {
    $group: { //Agrupa los resultados
      _id: "$id_local",
      rotulo: { $first: "$rotulo" },
      licencias_en_tramitacion: { $sum: 1 },
      terraza: { $first: "$terraza" }
    }
  },
  {
    $project: { //Selecciona los campos a mostrar
```

```
      _id: 0,
      id_local: "$_id",
      rotulo: 1,
      terraza: 1,
      licencias_en_tramitacion: 1
    }
  }
]
```

Figura 3: Locales y terrazas “En tramitación”

d - Clasificación por sección, división y epígrafe

```
[
  {
    $unwind: "$actividades" // Se desglosa el array de actividades
  },
  {
    $group: { //Se agrupa por seccion, division y epigrafe, y se maneja el caso vacio
      _id: {
        seccion: { $ifNull: ["$actividades.id_seccion", "Sin seccion"] },
        division: { $ifNull: ["$actividades.id_division", "Sin division"] },
        epigrafe: { $ifNull: ["$actividades.id_epigrafe", "Sin epigrafe"] },
        //
        desc_epigrafe:{ $ifNull: ["$actividades.desc_epigrafe", "Sin describir epigrafe"]}
      },
      totalLocales: { $sum: 1 }, // Se cuentan los locales en cada grupo
```

```
      locales: {
        $push: { // Se agregan los locales a un array dentro de cada grupo
          id_local: "$id_local",
          rotulo: "$rotulo",
          terraza: "$terraza"
        }
      }
    }
  },
  {
    $project: { // Se seleccionan los campos a mostrar
      _id: 0,
      seccion: "$_id.seccion",
      division: "$_id.division",
      epigrafe: "$_id.epigrafe",
      totalLocales: 1,
      locales: 1,
      desc_epigrafe: "$_id.desc_epigrafe"
    }
  },
  {
    $sort: { // Se ordena por total de locales
      totalLocales:-1
    }
  }
]
```

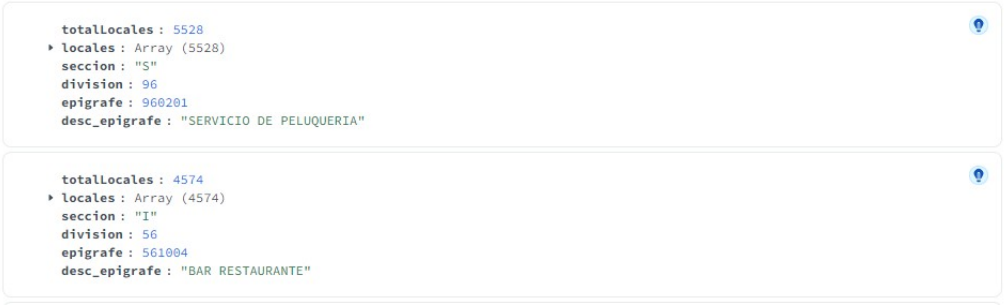

Figura 4: Clasificación de actividades

e - Actividad más frecuente por distrito y barrio

```
[
  {
    $unwind: "$actividades" // Se desglosa el array de actividades
  },
  {
    $group: { //Se agrupa por distrito, barrio y actividad
```

```
      _id: {
        desc_distrito_local: "$desc_distrito_local",
        desc_barrio_local: "$desc_barrio_local",
        desc_epigrafe: "$actividades.desc_epigrafe"
      },
      count: { $sum: 1 } // Cuenta la frecuencia de cada combinacion de distrito, barrio y actividad
    }
  },
  {
    $sort: { count: -1 } // Ordena los resultados por frecuencia de forma descendente
  },
  {
    $group: { // Se agrupa solo por distrito y barrio
      _id: {
        desc_distrito_local: "$_id.desc_distrito_local",
        desc_barrio_local: "$_id.desc_barrio_local"
      },
      actMasFrecuente: { $first: "$_id.desc_epigrafe" }, // Obtiene la actividad mas frecuente
      cuentaMasFrecuente: { $first: "$count" } // Obtiene la frecuencia de la actividad mas frecuente
    }
  },
  {
    $project: { //Se seleccionan los campos a mostrar
      _id: 0,
      distrito: "$_id.desc_distrito_local",
      barrio: "$_id.desc_barrio_local",
      actividad_mas_frecuente: "$actMasFrecuente",
      frecuencia: "$cuentaMasFrecuente"
    }
  },
    {
      $sort: {
        frecuencia: -1
      }
    }
]
```

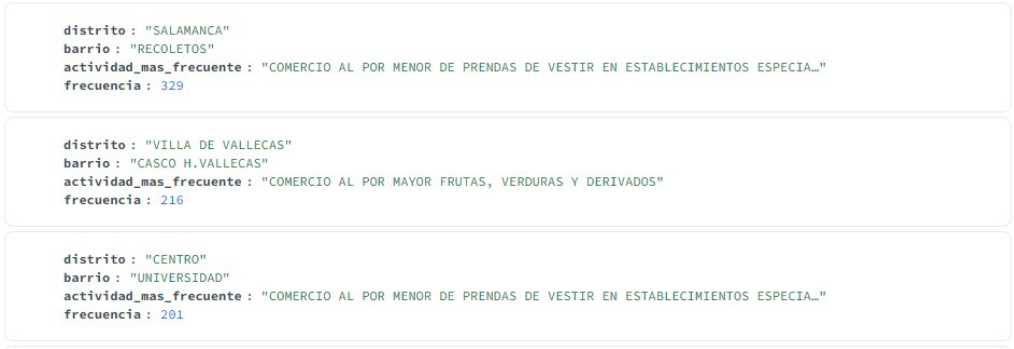

Figura 5: Actividad más frecuente por barrio

f - Cambio de horarios de Bares con Cocina en SALAMANCA

```
//filtrado
{
  "actividades.desc_epigrafe": "BAR CON COCINA",
  "desc_distrito_local": "SALAMANCA"
}
```

```
//actualizacion
{
  $set: {
    hora_apertura1: "11:00",
    hora_cierre1: "01:00"
  }
}
```

```
//Consulta para verificar cambio
[
  {
    $match: {
      "actividades.desc_epigrafe": "BAR CON COCINA",
      "desc_distrito_local": "SALAMANCA" // Filtrar por el distrito de
   Salamanca
    }
  },
  {
    $unwind: "$terraza" // Desglosa el array de terrazas para acceder a los
   horarios
  },
  {
    $project: { // Selecciona los campos a mostrar
      _id: 0,
      nombre_local: "$rotulo",
    distrito: "$desc_distrito_local",
      hora_apertura: "$hora_apertura1",
```

```
      hora_cierre: "$hora_cierre1"
    }
  }
]
```

2.3. Creación y uso de índices en MongoDB

Se crean tres tipos de índices distintos y luego se ejecutan consultas para evaluar su uso y desempeño:

a - Índice simple: Nombre del barrio

Creación del índice:

```
db.locales_madrid.createIndex({desc_barrio_local: 1})
```

Consulta para evaluar resultados:

```
//Consulta de locales con terraza en barrio que empiece con "A"
[
  {
    $match: {
      "terraza": { $exists: true, $not: { $size: 0 } }, // Filtra locales con terraza
      "desc_barrio_local": { $regex: "^A" } // Filtra barrios que empiezan con "A"
    }
  },
  {
    $group: {
      _id: "$desc_barrio_local", // Agrupa por barrio
      totalLocalesConTerraza: { $sum: 1 } // Cuenta locales con terraza en cada barrio
    }
  },
  {
    $sort: { _id: 1 } // Ordena alfabeticamente por nombre de barrio
  },
  {
    $project: { // Selecciona los campos a mostrar
      _id: 0,
      nombre_barrio: "$_id",
      totalTerrazas: "$totalLocalesConTerraza"
    }
  }
]
```

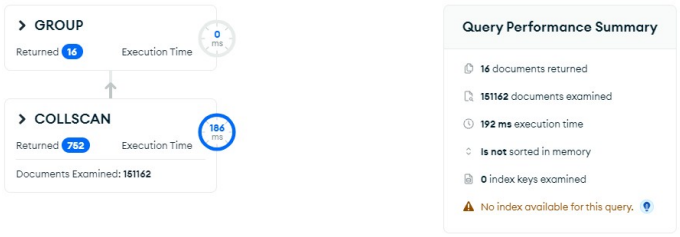

Figura 6: Rendimiento de la consulta sin el index simple

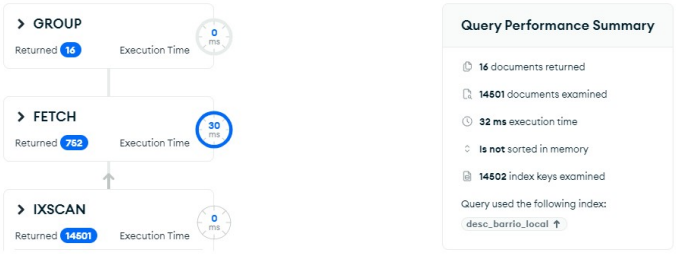

Figura 7: Rendimiento de la consulta con el index simple

Al comparar las figuras 6 y 7 se puede apreciar que el tiempo de la consulta baja de 192ms a 32 ms al usar el índice simple para desc_barrio_local.

b - Índice compuesto: Distrito y barrio Creación del índice:

```
db.locales_madrid.createIndex({desc_distrito_local: 1, desc_barrio_local:
    1})
```

Consulta para evaluar resultados:

```
//Consulta de locales con terraza en distrito que empiece con C y Barrio A
[
  {
    $match: {
      terraza: {
        $exists: true,
        $not: { $size: 0 }
      }, // Filtra locales con terraza
      desc_distrito_local: { $regex: "^C" }, // Filtra distritos que
    empiezan con "C"
      desc_barrio_local: { $regex: "^A" } // Filtra barrios que empiezan con
     "A"
    }
  },
  {
    $group: { //Agrupa por distrito y barrio
      _id: {
```

```
        distrito: "$desc_distrito_local",
        barrio: "$desc_barrio_local"
      },
      totalLocalesConTerraza: { $sum: 1 }
    }
  },
  {
    $sort: { "_id.distrito": 1, "_id.barrio": 1 } // Ordena por distrito y
    luego por barrio
  },
  {
    $project: { // Selecciona los campos a mostrar
      _id: 0,
      distrito: "$_id.distrito",
      barrio: "$_id.barrio",
      totalTerrazas: "$totalLocalesConTerraza"
    }
  }
]
```

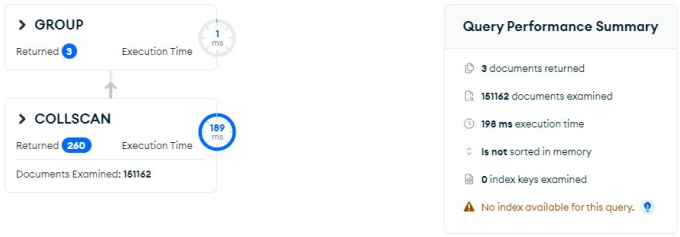

Figura 8: Rendimiento de la consulta sin el index compuesto

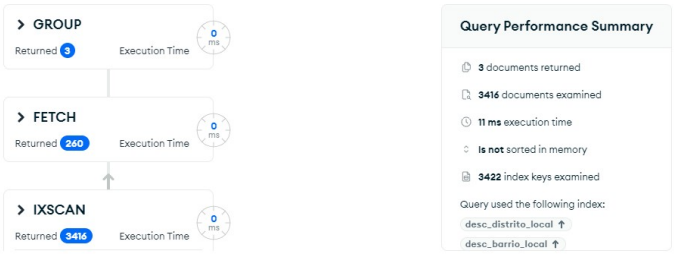

Figura 9: Rendimiento de la consulta con el index compuesto

Al comparar las figuras 8 y 9 se puede apreciar que el tiempo de la consulta baja de 198ms a 11 ms al usar el índice compuesto para desc_distrito_local y desc_barrio_local.

c - Índice de array: Descripción de actividad Creación del índice:

```
db.locales_madrid.createIndex({"actividades.desc_epigrafe": 1})
```

Consulta para evaluar resultados:

```
//Consulta de locales donde al menos una de sus actividades sea BAR CON
    COCINA
[
  {
    $match: {
      "actividades.desc_epigrafe": "BAR CON COCINA"
    }
  },
  {
    $count: "totalLocalesConBarCocina"
  }
]
```

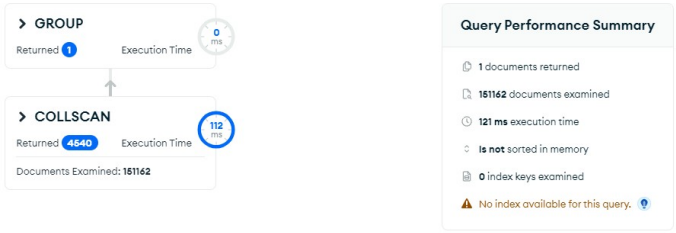

Figura 10: Rendimiento de la consulta sin el index de array

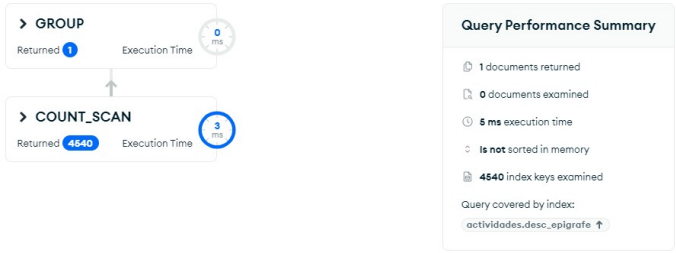

Figura 11: Rendimiento de la consulta con el index de array

Al comparar las figuras 10 y 11 se puede apreciar que el tiempo de la consulta baja de 121ms a 6 ms al usar el índice compuesto para desc_distrito_local y desc_barrio_local.

2.4. Modelo de datos: Extensión con alojamientos turísticos

Se incorporan datos de alojamientos turísticos de Airbnb en Madrid. Tomados de https://insideairbnb.com/madrid/. El fichero tiene 75 columnas, sin embargo para las consultas de la tarea muchas no son necesarias.

```
columns_to_keep = ["id", "name", "neighbourhood_cleansed", "
    neighbourhood_group_cleansed", "latitude", "longitude", "bathrooms", "bedrooms
    ", "amenities", "price", "number_of_reviews", "review_scores_rating"]
df_airbnb = df_airbnb[columns_to_keep]
```

Además nos aseguramos de limpiar los trailing y leading whitespaces de las columnas de texto:

```
for column in df_airbnb.columns:
    if df_airbnb[column].dtype == "object":
        df_airbnb[column] = df_airbnb[column].str.strip()
```

Limpiamos el formato de la columna "price". Eliminamos el "$", eliminamos la "," y convertimos la columna a float.

```
df_airbnb["price"] = df_airbnb["price"].str.replace("$","")
df_airbnb["price"] = df_airbnb["price"].str.replace(",","")
df_airbnb["price"] = df_airbnb["price"].astype(float)
```

Eliminamos las tildes y pasamos a mayúsculas los campos de neighbourhood_cleansed y neighbourhood_group_cleansed. Además reemplazamos los espacios antes y despues de los guiones:

```
df_airbnb["neighbourhood_cleansed"] = df_airbnb["neighbourhood_cleansed"].str.
    normalize('NFKD').str.encode('ascii', 'ignore').str.decode('utf-8').str.upper
    ()
df_airbnb["neighbourhood_group_cleansed"] = df_airbnb["
    neighbourhood_group_cleansed"].str.normalize('NFKD').str.encode('ascii', '
    ignore').str.decode('utf-8').str.upper
df_airbnb.neighbourhood_group_cleansed = df_airbnb.neighbourhood_group_cleansed.
    str.replace(" - ","-")
```

Cambiamos los nombres de las columnas para facilitar la redacción y lectura de las consultas:

```
nombres_columnas = ["id_airbnb", "nombre_airbnb", "barrio_airbnb", "
    distrito_airbnb", "latitud", "longitud", "numero_banos",
                    "numero_habitaciones", "servicios", "precio", "numero_resenas
    ", "puntuacion"]
df_airbnb.columns = nombres_columnas
```

Para insertar los datos a Mongo se van a manejar como una colección aparte en la misma base de datos. Para establecer las consultas que usan ambas colecciones, la relación se suele establecer por medio de los barrios o distritos de Madrid, ya que los valores se comparten entre ambas colecciones. Los servicios de cada alojamiento se anidan en un array.
Carga de datos a Mongo:

```
from pymongo import MongoClient
import ast

client = MongoClient("mongodb://localhost:27017/")
db = client["locales_madrid_mongo"]
```

```
alojamientos_collection = db["alojamientos"]

def preparar_datos(row):
  """Prepara un diccionario para insertar en MongoDB."""
  documento = row.to_dict()

  # Aseguramos que "servicios" sea un array
  if isinstance(documento["servicios"], str):
    documento["servicios"] = [s.strip() for s in documento["servicios"].strip("[]").replace('\"','').split(",")]
  elif not isinstance(documento["servicios"], list):  # Si no es ni string ni lista, lo dejamos vacio
    documento["servicios"] = []

  return documento

# Convertimos el DataFrame a una lista de diccionarios
datos_para_insertar = [preparar_datos(row) for _, row in df_airbnb.iterrows()]

# Insertamos los documentos
if datos_para_insertar:  # Verificamos que haya datos para insertar
  alojamientos_collection.insert_many(datos_para_insertar)
  print(f"Se insertaron {len(datos_para_insertar)} documentos en la coleccion.")
else:
  print("No hay datos para insertar.")

# Cerramos la conexion
client.close()
```

La estructura de los documentos en la colección “alojamientos” es la siguiente:


```
{
  "_id": {
    "$oid": "67a5d21accc82f73cdbb67d4"
  },
  "id_airbnb": 62423,
  "nombre_airbnb": "MAGIC ARTISTIC HOUSE IN THE CENTER OF MADRID",
  "barrio_airbnb": "JUSTICIA",
  "distrito_airbnb": "CENTRO",
  "latitud": 40.41884,
  "longitud": -3.69655,
  "numero_baños": 1.5,
  "numero_habitaciones": 1,
  "servicios": [
    "Books and reading material",
```

```
    "First aid kit",
    "Kitchen",
    "Essentials",
    ...
    ],
  "precio": 69,
  "numero_reseñas": 219,
  "puntuacion": 4.64,
  "id_barrio_alojamiento": 104,
  "id_distrito_alojamiento": 1
}
```

2.4.1. Consultas para evaluar el modelo

a - Total de alojamientos, locales y terrazas por distrito y barrio

```
[
  {
    $group: { //Agrupa los alojamientos por distrito y barrio
      _id: {
        distrito: "$distrito_airbnb",
        barrio: "$barrio_airbnb"
      },
      total_alojamientos: { $sum: 1 }
    }
  },

  {
    $lookup: { // Trae datos de locales_madrid usando el pipeline
      from: "locales_madrid",
      let: { distrito: "$_id.distrito", barrio: "$_id.barrio" },
      pipeline: [
        {
          $match: { //Busca que concidan el distrito y barrio con los
    alojamientos
            $expr: {
              $and: [
                { $eq: ["$desc_distrito_local", "$$distrito"] },
                { $eq: ["$desc_barrio_local", "$$barrio"] }
              ]
            }
          }
        },
        {
          $group: { //Cuenta locales y terrazas
```

```
            _id: null,
            total_locales: { $sum: 1 },
            total_terrazas: {
              $sum: {
                $cond: [
                  { $gt: [{ $size: { $ifNull: ["$terraza", []] } }, 0] },
                  1,
                  0
                ]
              }
            }
          }
        }
      ],
      as: "locales"
    }
  },
  {
    $unwind: "$locales" // Desglosa el array "locales"
  },
  {
    $project: { // Selecciona los campos mostrar
      _id: 0,
      distrito: "$_id.distrito",
      barrio: "$_id.barrio",
      total_alojamientos: 1,
      total_locales: "$locales.total_locales",
      total_terrazas: "$locales.total_terrazas"
    }
  },
  {
    $sort:{ // Ordena por distrito y barrio
      distrito: 1,
      barrio: 1
    }
  }
]
```

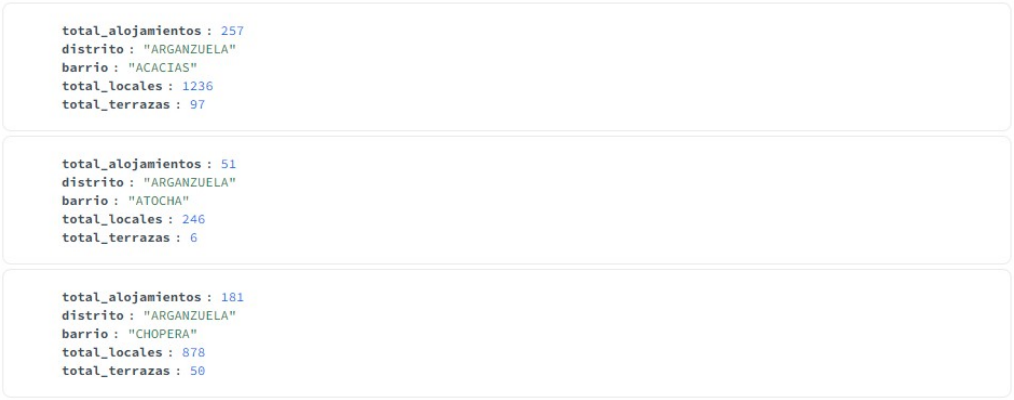

Figura 12: Alojamientos, locales y terrazas por barrio

b - 5 barrios con más alojamientos y terrazas con licencias concedidas hace 2 años o menos.

```
[
  {
    $group: {
      _id: {
        distrito: "$distrito_airbnb",
        barrio: "$barrio_airbnb"
      },
      total_alojamientos: {
        $sum: 1
      }
    }
  },
  {
    $lookup: {
      from: "locales_madrid",
      let: {
        distrito: "$_id.distrito",
        barrio: "$_id.barrio"
      },
      pipeline: [
        {
          $match: {
            $expr: {
              $and: [
                {
                  $eq: [
                    "$desc_distrito_local",
                    "$$distrito"
                  ]
                },
                {
                  $eq: [
```

```
                    "$desc_barrio_local",
                    "$$barrio"
                  ]
                }
              ]
            }
          }
        },
        {
          $unwind: "$licencias"
        },
        {
          $match: {
            // Se agrega la condicion para desc_situacion_licencia
            $expr: {
              $and: [
                {
                  $eq: [
                    "$licencias.desc_tipo_situacion_licencia",
                    "Concedida"
                  ]
                },
                {
                  $gte: [
                    "$licencias.Fecha_Dec_Lic",
                    "2023-02-08"
                  ]
                }
              ]
            }
          }
        },
        {
          $group: {
            _id: null,
            total_terrazas: {
              $sum: {
                $cond: [
                  {
                    $gt: [{$size: {$ifNull: ["$terraza",[]]}},0]},
                  1,
                  0
                ]
              }
            }
          }
        }
```

```
      ],
      as: "locales"
    }
  },
  {
    $unwind: "$locales"
  },
  {
    $project: {
      _id: 0,
      distrito: "$_id.distrito",
      barrio: "$_id.barrio",
      total_alojamientos: 1,
      total_terrazas: "$locales.total_terrazas"
    }
  },
  {
    $addFields: {
      suma_total: {
        $add: [
          "$total_alojamientos",
          "$total_terrazas"
        ]
      }
    }
  },
  {
    $sort: {
      suma_total: -1
    }
  },
  {
    $limit: 5
  }

]
```

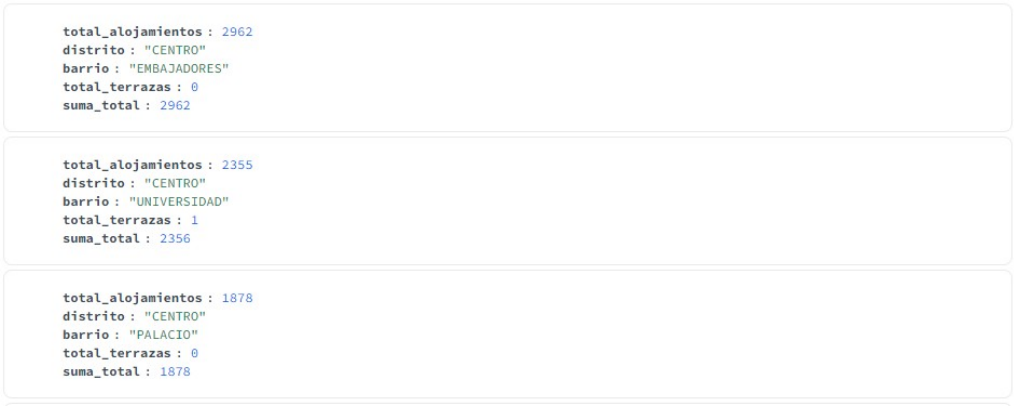

Figura 13: Cinco barrios con más alojamientos y terrazas concedidas hace menos de 2 años

c - Alojamientos con más reseñas por barrio y numero de locales con actividad comercial en dicho barrio

```
// Mas de 800 resenas
// Local que incluya COMERCIO AL POR MAYOR Y AL POR MENOR en desc_seccion

[
  {
    $match: // Se seleccionan alojamientos con mas de 800 resenas
      {
        numero_resenas: {
          $gte: 800
        }
      }
  },
  {
    $lookup: // Se traen los locales de los mismos barrios que los
    alojamientos
      {
        from: "locales_madrid",
        localField: "barrio_airbnb",
        foreignField: "desc_barrio_local",
        pipeline: [
          {
            $match: { // Se verifica que el local sea comercial
              "actividades.desc_seccion": {
                $regex:
                  /COMERCIO AL POR MAYOR Y AL POR MENOR/
              }
            }
          }
        ],
        as: "locales"
```

```
      }
  },
  {
    $project: // Se seleccionan los campos a mostrar
      {
        _id: 0,
        nombre_airbnb: 1,
        numero_resenas: 1,
        locales_comerciales: {
          $size: "$locales"
        }
      }
  }
]
```

Figura 14: Alojamientos con más reseñas por barrio, y locales comerciales en dicho barrio

d - Categorización de alojamientos por barrio, considerando precio, reseñas, numero de habitaciones y servicios

```
//precio
  //ECONOMICO <= 70
  //REGULAR >70 < 180
  //CARO >= 180
//resenas (puntuacion)
  //MALO <= 3
  //ACEPTABLE >3 <4
  //BUENO >= 4 <= 4,5
  //EXCELENTE > 4,5
//numero de habitaciones
  //PEQUENO 1
  //MEDIANO 3 5
  //GRANDE > 5
//servicios
  //SIN SERVICIOS 0
  //BASICO 10
  //REGULAR 25
  //MUCHOS SERVICIOS >25
```

```
19
20 [
21   {
22     $project: { // Se crean las categorias
23       precio_categoria: {
24         $switch: {
25           branches: [
26             { case: { $lte: ["$precio", 70] }, then: "ECONOMICO" },
27             { case: { $and: [{ $gt: ["$precio", 70] }, { $lte: ["$precio",
    180] }] }, then: "REGULAR" }
28           ],
29           default: "CARO"
30         }
31       },
32       puntuacion_categoria: {
33         $switch: {
34           branches: [
35             { case: { $lt: ["$puntuacion", 3] }, then: "MALO" },
36             { case: { $and: [{ $gte: ["$puntuacion", 3] }, { $lte: ["
    $puntuacion", 4] }] }, then: "ACEPTABLE" },
37             { case: { $and: [{ $gt: ["$puntuacion", 4] }, { $lte: ["
    $puntuacion", 4.5] }] }, then: "BUENO" }
38           ],
39           default: "EXCELENTE"
40         }
41       },
42       habitaciones_categoria: {
43         $switch: {
44           branches: [
45             { case: { $eq: ["$numero_habitaciones", 1] }, then: "PEQUENO" },
46             { case: { $and: [{ $gte: ["$numero_habitaciones", 3] }, { $lte:
    ["$numero_habitaciones", 5] }] }, then: "MEDIANO" }
47           ],
48           default: "GRANDE"
49         }
50       },
51       servicios_categoria: {
52         $switch: {
53           branches: [
54             { case: { $eq: [{ $size: "$servicios" }, 0] }, then: "SIN
    SERVICIOS" },
55             { case: { $and: [{ $gte: [{ $size: "$servicios" }, 1] }, { $lte:
     [{ $size: "$servicios" }, 10] }] }, then: "BASICO" },
56             { case: { $and: [{ $gte: [{ $size: "$servicios" }, 10] }, { $lte
    : [{ $size: "$servicios" }, 25] }] }, then: "REGULAR" }
57           ],
58           default: "MUCHOS"
```

```
        }
      },
      barrio_airbnb: 1,
      nombre_airbnb: 1
    }
  },
  {
    $group: { // Se agrupan con respecto a las categorias creadas
      _id: {
        barrio: "$barrio_airbnb",
        precio: "$precio_categoria",
        puntuacion: "$puntuacion_categoria",
        habitaciones: "$habitaciones_categoria",
        servicios: "$servicios_categoria"
      },
      count: { $sum: 1 },
      nombres_airbnb: { $addToSet: "$nombre_airbnb" } // Usamos $addToSet
   para evitar nombres repetidos
    }
  },
  {
    $group: {
      _id: "$_id.barrio",
      categorias: {
        $push: {
          precio: "$_id.precio",
          puntuacion: "$_id.puntuacion",
          habitaciones: "$_id.habitaciones",
          servicios: "$_id.servicios",
          count: "$count",
          nombres_airbnb: "$nombres_airbnb" // Incluimos el array de nombres
        }
      }
    }
  },
  { $sort: { "_id": 1, "categorias.precio": 1, "categorias.puntuacion": 1, "
   categorias.habitaciones": 1, "categorias.servicios": 1 } }
]
```

Figura 15: Categorización de alojamientos

3. Visualización de datos: Modelo de grafo

Se presenta un pequeño ejemplo del uso de un modelo de grafo en Neo4j, basado en los datos utilizados durante la tarea.

La estructura general de los nodos viene dada por:

- Local: Nombre, tipo de acividad, horario, licencia
- Terraza: Nombre, capacidad, estado de licencia
- Alojamiento: Nombre, precio, número de habitaciones, reseñas
- Barrio: Nombre

El modelo considera las siguientes relaciones:

- Ubicado_en: Conecta Local, Terraza y Alojamiento con su Barrio. Tiene el atributo *distancia*.
- Cercano_a: Conecta Local, Terraza y Alojamiento entre sí si están a menos de cierta distancia definida. Tiene el atributo *distancia*.
- Relacionado_con: conecta nodos de la misma categoría que comparten atributos comunes, como tipo de actividad o precio

3.1. Creación del grafo

Inicialmente creamos los nodos:

Nodos de barrios

```
CREATE (b1:Barrio {nombre: "Salamanca"})
CREATE (b2:Barrio {nombre: "Chamberi"})
CREATE (b3:Barrio {nombre: "Centro"})
```

Nodos de locales

```
CREATE (l1:Local {nombre: "Cafeteria Goya", tipo_actividad: "Restauracion",
    horario: "08:00-22:00", licencia: "Concedida"})
CREATE (l2:Local {nombre: "Libreria Central", tipo_actividad: "Cultura", horario:
     "10:00-20:00", licencia: "En tramite"})
```

Nodos de terrazas

```
CREATE (t1:Terraza {nombre: "Terraza Sol", capacidad: 30, estado_licencia: "Concedida"})
CREATE (t2:Terraza {nombre: "Terraza Norte", capacidad: 20, estado_licencia: "En tramite"})
```

Nodos de alojamiento

```
CREATE (a1:Alojamiento {nombre: "Airbnb Retiro", precio: 120, numero_habitaciones: 2, resenas: 45, servicios: ["WiFi", "Cocina"]})
CREATE (a2:Alojamiento {nombre: "Apartamento Goya", precio: 80, numero_habitaciones: 1, resenas: 0, servicios: ["Aire acondicionado"]})
```

Una vez creados los nodos establecemos las relaciones entre ellos:

Relacionamos locales, terrazas y alojamientos con barrios mediante "Ubicado_en":

```
MATCH (b:Barrio {nombre: "Salamanca"}), (l:Local {nombre: "Cafeteria Goya"})
CREATE (l)-[:Ubicado_en {distancia: 0.5}]->(b)
MATCH (b:Barrio {nombre: "Centro"}), (a:Alojamiento {nombre: "Airbnb Retiro"})
CREATE (a)-[:Ubicado_en {distancia: 0.3}]->(b)
```

Ahora relacionamos entidades cercanas entre sí mediante "Cercano_a":

```
MATCH (l:Local {nombre: "Cafeteria Goya"}), (t:Terraza {nombre: "Terraza Sol"})
CREATE (l)-[:Cercano_a {distancia: 0.2}]->(t)
MATCH (a:Alojamiento {nombre: "Airbnb Retiro"}), (l:Local {nombre: "Libreria Central"})
CREATE (a)-[:Cercano_a {distancia: 0.1}]->(l)
```

Finalmente relacionamos nodos que cuentan con atributos comunes:

```
MATCH (l:Local {tipo_actividad: "Restauracion"}), (t:Terraza {nombre: "Terraza Sol"})
CREATE (l)-[:Relacionado_con {motivo: "Ambiente de comida"}]->(t)
```

Como resultado obtenemos el grafo mostrado a continuación:

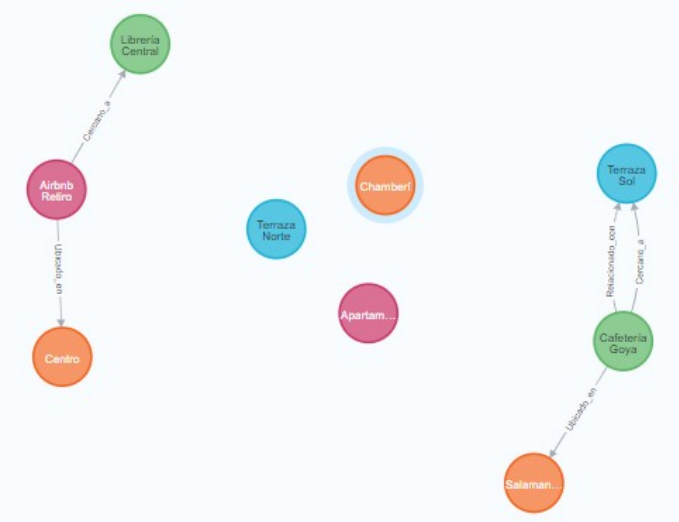

Figura 16: Grafo de comercio y alojamiento en Madrid

3.2. Consultas de validación

a - Locales y terrazas en Salamanca

```
MATCH (b:Barrio {nombre: "Salamanca"})<-[:Ubicado_en]-(n)
WHERE n:Local OR n:Terraza
RETURN n.nombre AS nombre, labels(n) AS tipo
```

Figura 17: Locales y terrazas en Salamanca

b - Alojamientos cuyo precio supere 100 EUR

```
MATCH (a:Alojamiento)
WHERE a.precio > 100
RETURN a.nombre AS nombre, a.precio AS precio
```

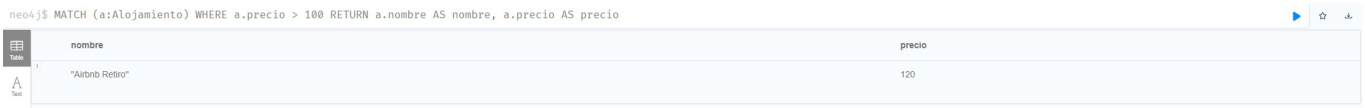

Figura 18: Alojamientos cuyo precio supere 100 EUR

c - Barrios donde los alojamientos no cuentan con dormitorios

```
MATCH (a:Alojamiento)-[:Ubicado_en]->(b:Barrio)
WHERE a.numero_habitaciones = 0
RETURN b.nombre AS barrio, COUNT(a) AS total_alojamientos
```

Figura 19: Barrios donde los alojamientos no cuentan con dormitorios

d - Barrios donde los alojamientos no cuentan con reseñas

```
MATCH (a:Alojamiento)-[:Ubicado_en]->(b:Barrio)
WHERE a.resenas = 0
RETURN b.nombre AS barrio, COUNT(a) AS total_alojamientos
```

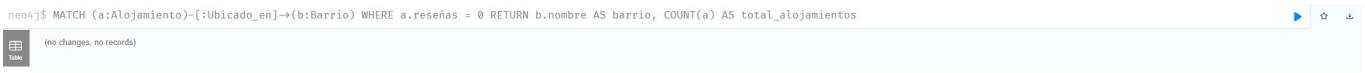

Figura 20: Barrios donde los alojamientos no cuentan con reseñas