Trabajo Práctico Obligatorio - Base de Datos II

Ahumada, Manuel - 64677
Gonzalez Cornet, Josefina - 64550
Ruckauf, Federico Ignacio - 64356
Salinas, Pedro - 64388

13/11/2025

Índice

Índice..... 2
1. Introducción..... 3
2. Desarrollo..... 3
3. Interacción..... 4
4. Consultas..... 4
    4.1 Query 1..... 4
    4.2 Query 2..... 5
    4.3 Query 3..... 5
    4.4 Query 4..... 6
    4.5 Query 5..... 7
    4.6 Query 6..... 7
    4.7 Query 7..... 8
    4.8 Query 8..... 9
    4.9 Query 9..... 9
    4.10 Query 10..... 11
    4.11 Query 11..... 11
    4.12 Query 12..... 12
    4.13 Query 13..... 13
    4.14 Query 14..... 15
    4.15 Query 15..... 16

1. Introducción

El siguiente trabajo práctico consiste en el desarrollo de un sistema de Backoffice que permite gestionar información relacionada con clientes, pólizas, siniestros, vehículos y agentes de una compañía de seguros.

2. Desarrollo

Para el mismo, se realizó una API que contiene queries con las cuales se puede obtener información específica de dichos datos. Los datos han sido persistidos utilizando la estrategia de persistencia políglota, utilizando 2 bases de datos no relacionales (NoSQL): MongoDB y Neo4J. A continuación se detalla por qué han sido elegidas:

- MongoDB: Cada archivo es persistido en una colección con el mismo nombre, donde cada fila es un documento con los atributos del header y los valores de las celdas encontradas en el csv. Las queries simples, que no requieren de relaciones entre distintas entidades son resueltas mediante el lenguaje de consulta de MongoDB.
- Neo4J: Esta base de datos se encarga de poder resolver de forma eficiente consultas que involucran relaciones conceptuales entre distintas entidades, es decir distintos archivos CSV. Gracias al poder de consulta de Cypher, y las características de una base de datos de grafos, se logran hacer queries simples que resuelven relaciones entre entidades.

Al utilizar ambas bases de datos obtenemos la mayor eficiencia de consulta, con MongoDB resolvemos queries simples directamente utilizando JSON y de forma rápida; y con Neo4J se soluciona el problema de la dificultad de consultas de agregación de Mongo para queries complejas entre entidades.

Dada a la baja cantidad de datos, se decidió no implementar índices adicionales. Sin embargo, en caso de desear escalar el sistema y aumentar en órdenes de magnitud la cantidad de datos, si sería de gran utilidad la implementación de los mismos por cuestiones de eficiencia.

3. Interacción

Para poder realizar las queries de forma sencilla, se presenta una API documentada mediante SwaggerUI. Esto hace que al hacer mediante un browser un GET a /docs se presente una pantalla con todas las queries y una interfaz de usuario que permite ejecutarlas de forma sencilla. Lea el readme presente en el repositorio para las instrucciones completas de uso de este trabajo práctico.

4. Consultas

A continuación se expone que datos y en qué orden devuelve cada consulta:

4.1 Query 1

Clientes activos con sus pólizas vigentes.

Método de consulta:

- GET /queries/query1

Salida:

- Nombre del cliente como “nombre”.
- Apellido del cliente como “apellido”.
- Arreglo con los números de pólizas vigentes correspondientes al cliente como “polizas_vigentes”, cada una identificada por su “nro_poliza”, ordenados ascendentemente.

Ejemplo de salida:

```
[
  {
    "nombre": "Laura",
    "apellido": "Gómez",
    "polizas_vigentes": [
      {
```

```
        "nro_poliza": "POL1001"
      }
    ]
  },...
]
```


4.2 Query 2

Siniestros abiertos con tipo, monto y cliente afectado.

Método de consulta:

- GET /queries/query2

Salida:

- Tipo de siniestro como “tipo”
- Monto estimado necesario para cubrir el siniestro como “monto”
- Nombre del cliente como “nombre”
- Apellido del cliente como “apellido”

Ejemplo de salida:


```
[
  {
    "tipo": "Accidente",
    "monto": "500000",
    "nombre": "Laura",
    "apellido": "Gómez"
  },...
]
```


4.3 Query 3

Vehículos asegurados con su cliente y póliza.

Aclaración: los datos son ordenados por patente ascendentemente.

Método de consulta:

- GET /queries/query3

Salida:

- Número de la póliza que cubre al vehículo como “nro_poliza”
- Devuelve patente, cliente y nro_poliza de todos los vehículos asegurados:
- Nombre del cliente como “nombre”
- Apellido del cliente como “apellido”
- Patente del vehículo como “patente”

Ejemplo de salida:

```
[
  {
    "nombre": "Laura",
    "apellido": "Gómez",
    "patente": "AE123CD",
    "nro_poliza": "POL1001"
  },...
]
```

4.4 Query 4

Clientes sin pólizas activas.

Método de consulta:

- GET /queries/query4

Salida:

- Nombre del cliente como “nombre”
- Apellido del cliente como “apellido”

Ejemplo de salida:

```
[
  {
    "nombre": "Sofía",
    "apellido": "López"
  },...
]
```

4.5 Query 5

Agentes activos con cantidad de pólizas asignadas.

Método de consulta:

- GET /queries/query5

Salida:

- Nombre del cliente como “nombre”
- Apellido del cliente como “apellido”
- Cantidad de pólizas vinculadas al cliente como “cant_polizas”

Ejemplo de salida:

```
[
  {
    "nombre": "María",
    "apellido": "Martínez",
    "polizas": 44
  },...
]
```

4.6 Query 6

Pólizas vencidas con el nombre del cliente.

Método de consulta:

- GET /queries/query6

Salida:

- Número de póliza como “nro_poliza”
- Nombre del cliente como “cliente”
- Apellido del cliente como “apellido”

Ejemplo de salida:

```
[
  {
    "nro_poliza": "POL1003",
    "nombre": "Martín",
    "apellido": "Pérez"
  },...
]
```

4.7 Query 7

Top 10 clientes por cobertura total.
Aclaración: dentro de ese top 10 el orden de desempate si la cobertura es igual es ascendente por apellido.

Método de consulta:

- GET /queries/query7

Salida:

- Total acumulado en coberturas por el cliente como “total_cobertura”
- Nombre del cliente como “cliente”
- Apellido del cliente como “apellido”

Ejemplo de salida:

```
[
  {
    "total_cobertura": 2500000,
    "nombre": "Pacífica",
    "apellido": "Barreda"
  },...
]
```

4.8 Query 8

Siniestros tipo “Accidente” del último año.

Método de consulta:

- GET /queries/query8

Salida:

- Identificador del siniestro como “id_siniestro”

Ejemplo de salida:

```
[
  {
    "id_siniestro": "9001"
  },
]
```

4.9 Query 9

Vista de pólizas activas ordenadas por fecha de inicio.
Aclaración: la vista se define a la hora de importar los datos a la base de datos Mongo. La ejecución de la query simplemente realiza el equivalente a un `select *` sobre la vista. Al no ser materializadas las vistas, esta implementación no causa ineficiencias en caso de que nunca se ejecute la query.

Método de consulta:

- GET /queries/query9

Salida:

- Object id propio de MongoDB como “_id”
- Número de la póliza como “nro_poliza”
- Identificador del cliente correspondiente como “id_cliente”
- Tipo de póliza como “tipo”
- Fecha de inicio de validez de la póliza como “fecha_inicio”
- Fecha de fin de validez de la póliza como “fecha_final”
- Prima mensual como “prima_mensual”
- Cobertura total como “cobertura_total”
- Identificador del agente correspondiente como “id_agente”
- Estado de la póliza como “estado”

Aclaración: al tratarse de una vista, se decidió retornar todos los datos completos para no intervenir con la idea que se haría el usuario sobre los datos existentes en la tabla “pólizas”.

Ejemplo de salida:


```
[
  {
    "_id": "6914df1cd50838323ecd0d93",
    "nro_poliza": "POL1113",
    "id_cliente": "113",
    "tipo": "Auto",
    "fecha_inicio": "1/1/2024",
    "fecha_fin": "1/1/2025",
    "prima_mensual": "12000",
    "cobertura_total": "800000",
    "id_agente": "104",
    "estado": "Activa"
```

```
  },...
]
```

4.10 Query 10

Pólizas suspendidas con estado del cliente.

Método de consulta:

- GET /queries/query10

Salida:

- Número de la póliza correspondiente como “nro_poliza”
- Estado del cliente como “estado_cliente”

Ejemplo de salida:

```
[
  {
    "nro_poliza": "POL1005",
    "cliente": {
      "estado": "Activo"
    }
  },...
]
```

4.11 Query 11

Clientes con más de un vehículo asegurado.

Método de consulta:

- GET /queries/query11

Salida:

- Nombre del cliente como “nombre”

- Apellido del cliente como “apellido”
- Cantidad de vehículos asegurados por el cliente como “cant_vehiculos”
- Estado del cliente como “estado_cliente”

Ejemplo de salida:

```
[
  {
    "nombre": "Laura",
    "apellido": "Gomez",
    "vehiculos": 3
  }
]
```

Aclaración: con el dataset original provisto por la cátedra la query da como resultado un arreglo vacío ya que en un principio no hay clientes con más de un vehículo asegurado.

4.12 Query 12

Agentes y cantidad de siniestros asociados.

Método de consulta:

- GET /queries/query12

Salida:

- Nombre del agente como “nombre”
- Apellido del agente como “apellido”
- Cantidad de siniestros asociados al cliente como “siniestros”

Ejemplo de salida:

```
[
  {
    "nombre": "Juan",
```

```
    "apellido": "Torres",
    "siniestros": 10
  },...
]
```

4.13 Query 13

ABM de clientes.

Esta query se llevó a cabo a través de 3 endpoints distintos que presentan distintas funcionalidades para clientes: crear, modificar y eliminar.

Método de consulta:

- POST /queries/crear-clientes

Request body:

```
{
  "id_cliente": "CLI-100",
  "nombre": "Roberto",
  "apellido": "Ramirez",
  "dni": "33111222",
  "email": "rober@correo.com",
  "telefono": "+54 11 4444-5555",
  "direccion": "Av. Siempre Viva 123",
  "ciudad": "Buenos Aires",
  "provincia": "Buenos Aires",
  "activo": "True"
}
```

Response:

```
{
```

```
  "success": true,
  "neo": "CLI-100",
  "mongo": "6914e6139554f9b15e2a2229"
}
```


Método de consulta:

- PUT /queries/modificar-cliente/:id

Request body:


```
{
  "nombre": "Manu"
}
```


Response:


```
{
  "success": true,
  "neo": "CLI-100",
  "mongo": 1
}
```


Método de consulta:

- DELETE /queries/borrar-cliente/:id

Response:


```
{
  "success": true,
  "mongo": {
    "deletedCliente": 1,
```

```
    "deletedPolizas": 0,
    "deletedSiniestros": 0
  },
  "neo": {
    "deletedCliente": 1,
    "deletedPolizas": 0,
    "deletedSiniestros": 0
  }
}
```


4.14 Query 14

Alta de nuevos siniestros.

Método de consulta:

- POST /queries/crear-siniestro

Request body:


```
{
  "id_siniestro": "SIN-200",
  "nro_poliza": "PZ-5001",
  "fecha": "25/05/2024",
  "tipo": "Accidente",
  "monto_estimado": 320000,
  "descripcion": "Colisión en autopista",
  "estado": "Abierto"
}
```


Response:

```
{
  "success": true,
  "neo": "SIN-200",
  "mongo": "6914e72b9554f9b15e2a222c"
}
```


4.15 Query 15

Emisión de nuevas pólizas (validando cliente y agente).

Método de consulta:

- POST /queries/crear-siniestro

Request body:


```
{
  "nro_poliza": "PZ-5001",
  "id_cliente": "CLI-100",
  "tipo": "Hogar",
  "fecha_inicio": "10/03/2024",
  "fecha_fin": "10/03/2025",
  "prima_mensual": 18000,
  "cobertura_total": 5000000,
  "id_agente": "101",
  "estado": "Activa"
}
```


Response:


```
{
  "success": true,
```

```
  "neo": "PZ-5001",
  "mongo": "6914e71f9554f9b15e2a222b"
}
```