MongoDB para el análisis de datos de calidad del aire: diseño y consulta sobre series temporales

Ana Cortazzo
ana.cortazzo@fing.edu.uy

Pablo Schiavone
pablo.shiavone@fing.edy.uy

Maestría en ciencia de datos y aprendizaje automático
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay

***Resumen*—Esta investigación se enfoca en analizar el uso de MongoDB como motor de base de datos para el manejo de series temporales con datos de calidad del aire en Montevideo. El estudio implica una revisión de trabajos relacionados, un análisis exploratorio del conjunto de datos, definición de posibles diseños de bases de datos y comparación empírica de posibles consultas para la búsqueda de patrones. Luego del estudio se puede concluir que MongoDB es una herramienta adecuada para el manejo de series temporales, y la organización de las colecciones por contaminantes ofrece ventajas a la hora de realizar consultas respecto a los demás diseños.**

I. Introducción

Existen muchas aplicaciones en la actualidad que requieren almacenar y analizar grandes volúmenes de datos con fecha y hora (*timestamp)*, desde aplicaciones de compras online, hasta todo el espectro de aplicaciones de Internet de las cosas (*Internet of things - IoT)*. En particular, los datos provenientes de sensores tienden a tener un gran volumen y alta frecuencia. La selección del mejor motor de base de datos para las aplicaciones de IoT es un tema de investigación actual dado que se requiere capacidad para trabajar con variedad de datos y esquemas flexibles, deben ser escalables y tener características adecuadas para trabajar con serie de datos temporales (*time-series data)* (MongoDB, 2023). En este sentido, las bases de datos documentales, en particular MongoDB, surgen como una alternativa interesante ya que cumplen con los requerimientos necesarios.

Este trabajo tiene como objetivo estudiar las posibilidades de MongoDB como motor para trabajar con series de datos temporales provenientes de medición de calidad del aire en la ciudad de Montevideo. Para esto se analizarán diferentes esquemas documentales y organización en colecciones, y a partir de ellos se realizarán diferentes consultas que permitan definir, de forma cualitativa, el diseño más adecuado. El trabajo se desarrolla en varias etapas, comenzando por una revisión bibliográfica de trabajos relacionados (Sección II), luego se presentan las características de los datos y una breve exploración de los mismos (Sección III), a partir de eso de presentan los diseños propuestos (Sección IV) y en base a los diseños, las consultas realizadas sobre la base de datos (Sección V); finalmente se presentan las conclusiones y trabajos futuros (Sección VI).

II. Trabajos relacionados

Las bases de datos documentales son un tipo de bases de datos no relacional (también conocidas como NoSQL), se organizan en *colecciones* que están pobladas por documentos, generalmente en formato JSON o XML (Harrison, 2015). Los documentos dentro de una colección pueden tener diferentes campos, esto configura diferentes esquemas. El diseño de un esquema adecuado implica tomar decisiones sobre cómo se van a representar los elementos de la realidad y las relaciones entre ellos. MongoDB es una base de datos documental que internamente utiliza una variante codificada en binario de JSON llamada BSON (Harrison, 2015). MongoDB ofrece flexibilidad en la definición del esquema a ser implementado junto con la posibilidad de elegir documentos embebidos o referenciados. Sumado a esto, MongoDB ofrece funcionalidades específicas para trabajar con serie de datos temporales, incluyendo drivers para Python y R, junto con MongoDB Charts que proporciona una forma fácil y rápida para visualizar los datos (MongoDB, 2019).

Varias investigaciones se han llevado a cabo sobre implementación de diferentes esquemas documentales para tratar con series temporales. Ramesh et al. estudian diferentes esquemas de modelado de datos para almacenar datos de series de tiempo discretas en Cassandra y MongoDB. Se describen tres modelos principales: almacenar los datos para un único anemómetro por fila, columnas válidas solo por un período de tiempo específico o número constante de entradas, y limitar el tamaño de la fila dividiendo los datos en múltiples filas. Los autores concluyen que tanto Cassandra como MongoDB son opciones adecuadas para almacenar datos de series temporales, cada una con sus ventajas específicas; Cassandra es recomendada cuando se requiere una alta escalabilidad y un acceso rápido a los datos, mientras que MongoDB ofrece flexibilidad en el esquema y un lenguaje de consulta más rico.

Mehmood et al. presentan un enfoque general para modelar los aspectos temporales de los datos de los sensores. Los autores desarrollan un prototipo para la plataforma en tiempo real MongoDB y discuten los retos y las decisiones de modelado de datos temporales. Sumando información espacial a los datos temporales, Widyani et al. estudian el proceso de mapeo de datos de desastres espacio-temporales en MongoDB y destacan la elección de MongoDB debido a su capacidad para manejar

grandes volúmenes de datos y cumplir con los requisitos de expresión de datos en un entorno espacio-temporal.

Un estudio comparativo entre la base de datos relacional MySQL y MongoDB como base de datos no relacional es realizado por Eyada et al.. Los autores definen dos tipos de esquema diferente (referencia e híbrido) y examinan el impacto del aumento de las cargas de trabajo y proponen un modelo de predicción a partir de análisis estadístico de los datos medidos en los experimentos realizados. Los resultados mostraron que el aumento de las cargas de trabajo conduce a una considerable pérdida de rendimiento de MySQL frente MongoDB y el modelo híbrido es mejor que el modelo de referencia.

No todos los trabajos relacionados fueron favorables a MongoDB. Un caso es el estudio realizado por Makris et al. que compararon el rendimiento en tiempo de respuesta entre MongoDB y PostgreSQL con la extensión PostGIS. Realizaron pruebas utilizando consultas espaciales y temporales en conjuntos de datos proporcionados por MarineTraffic. Los resultados mostraron que PostgreSQL superó a MongoDB en todas las consultas, con una mejora promedio de velocidad de aproximadamente 2 veces en la primera consulta, 4 veces en la segunda y 4.2 veces en la tercera, en un clúster de 5 nodos. Además, las implementaciones de clúster de réplicas y replicación en streaming superaron la implementación de un solo nodo en ambos sistemas.

El estudio realizado por Di Martino et al. comparó el rendimiento en términos de ingesta, recuperación y espacio de almacenamiento requerido para datos de IoT en motores Apache Cassandra, MongoDB y InfluxDB. En este trabajo MongoDB ofreció el mejor rendimiento en consultas sobre atributos indexados no temporales pero InfluxDB resultó ser en promedio la solución más equilibrada, superando a ambos competidores en aspectos de almacenamiento y proporcionando el mejor rendimiento en ingesta y consultas basadas en series temporales

III. Análisis exploratorio del conjunto de datos

Para este trabajo se utilizaron datos abiertos de la red de monitoreo de la calidad del aire de Montevideo[1]. En el análisis inicial se determinó la calidad de los datos, reconociendo formato y contenido, cantidad de valores valores nulos y se realizaron algunas visualizaciones preliminares[2]. Los datos se encuentran es formato `csv` y se cuenta con datos históricos (2003-2018 en un único archivo) y datos recientes en archivos anuales (2019-2022). Además se tiene la información (*metadata*) asociada a los contaminantes, las estaciones y los métodos de medición.

La red de monitoreo cuenta con 18 estaciones distribuidas en 7 zonas de la ciudad[3] y recolecta datos de 7 contaminantes diferentes que son presentados en detalle en la Tabla I. No todas las estaciones se encuentran activas en todos los años y no tienen datos de los mismos contaminantes, a partir del año 2019 solo se tiene datos de los contaminantes NO_2, O_3, $PM2$ y $PM10$. En este trabajo serán utilizados los siguientes campos:

- `pollutant_id`: identificación del contaminante
- `pollutant_value`: medida del contaminante
- `pollutant_unit`: unidad de medida ($\mu g/m^3$)
- `date`: fecha y hora de la medida (la frecuencia varía según el contaminante)
- `station_id`: identificador de la estación de medida
- `X` y `Y`: coordenadas geográficas de la estación de medida
- `method_id`: método de medición del contaminante

Los valores nulos en el campo `pollutant_value` son determinantes para el análisis, los datos cuentan con un gran número de valores nulos, distribuidos temporalmente, por estación y por contaminante. De forma general podemos afirmar que no existe un registro anual completo por contaminante en ninguna de las estaciones de medición, presentando en algunos casos falta de registros por más de un mes.

Para seleccionar los datos para el trabajo se determinó la cantidad de días al año que se cuenta con alguna medición y en base a eso se seleccionaron los datos desde el 2016 en adelante, que cuentan con datos durante todos los días del año[4]. Para determinar una estrategia adecuada de diseño, con el objetivo de mejorar la calidad del análisis, se realizaron diferentes visualizaciones que permitan ver tendencias anuales o mensuales.

En la Figura 1 se presenta, a modo de ejemplo, una visualización de la variación de promedios mensuales a lo largo de los años para el contaminante $PM2$ donde se puede observar que en los meses de mayo a agosto presenta los valores más elevados en promedio, aunque hay una gran variación respecto a la media en ese período. Los valores atípicos (*outliers*) superiores corresponden al año 2019. La Figura 2 presenta un detalle de los datos de $PM2$ discriminados por estación, donde se puede observar que para ninguna estación la serie temporal está completa, siendo la estación de Ciudad Vieja (`UYMVD_E1`) la que presenta la serie más completa. Este análisis se repitió para todos los contaminantes y se puede encontrar completo en el repositorio del trabajo.

Tabla I: Detalle de contaminantes

Contaminante	Explicación
$PM2$	Particulado menor de 2,5 μm de diám.
$PM10$	Particulado menor de 10 μm de diám.
HN	Humo Negro (black smoke)
PTS	Material particulado total
SO_2	Dióxido de azufre
NO_2	Dióxido de nitrógeno
O_3	Ozono

[1]Datos publicados en el Catálogo Nacional de Datos Abiertos.

[2]El análisis exploratorio se realizó utilizando *Jupyter Notebook* y se puede encontrar completo en el repositorio de GitLab.

[3]Ciudad Vieja, Centro, Tres Cruces, Curva de Maroñas, Portones de Carrasco, La Teja, Colón, Goes, Prado.

[4]Esto no quiere decir que los datos correspondan al mismo contaminante o a la misma estación, pero se garantiza que hay algún dato registrado cada día del año.

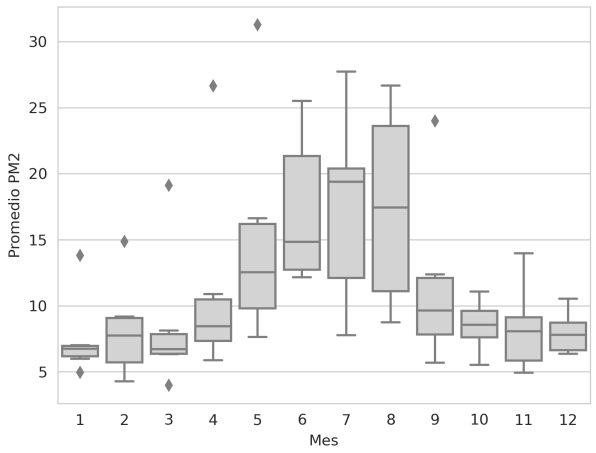

Figura 1: Variación del promedio mensual de PM2 en el período 2016-2022.

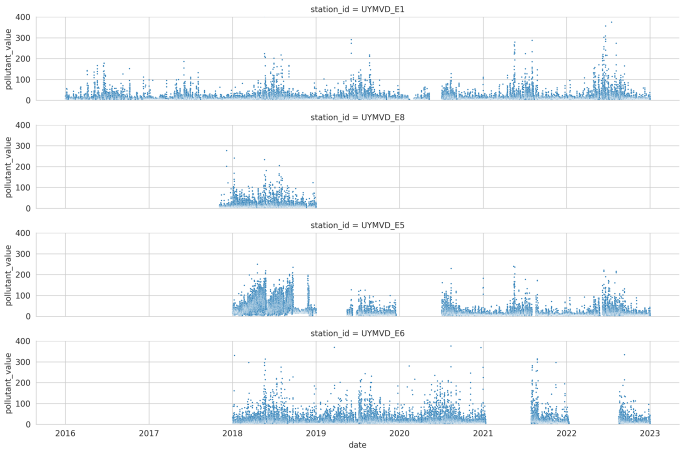

Figura 2: Detalle de los datos de PM2 en el período 2016-2022 discriminados por estación.

Los valores faltantes (indicados como `NA` en el conjunto de datos) se deben a diferentes factores[5] y pueden influir negativamente en el análisis de los datos. Existen diferentes estrategias para el tratamiento de los datos no disponibles como *imputación* (completar los datos que faltan basados en observaciones sobre el conjunto de datos), *Interpolación* (utilizar puntos de datos vecinos para estimar el valor que falta) o la *eliminación de periodos de tiempo afectados* (cuando se opta por no utilizar los periodos de tiempo en los que faltan datos) (Nielsen, 2019). En este caso imputar valores en períodos muy largos puede no ser adecuado ya que modificaría sustancialmente las estadísticas de los datos. El objetivo de este trabajo no es realizar un análisis profundo ni predicciones en base a los datos, por lo que se opta por eliminar los períodos de tiempo con valores `NA`.

En este trabajo se consideraron los contaminantes $PM2$, $PM10$, NO_2 y O_3 ya que son los que cuentan con datos más completos a lo largo de los años, y las estaciones consideradas son aquellas que tienen datos de estos contaminantes.

[5]Mantenimientos, verificaciones automáticas de cero, calibraciones, presencia de interferencias, roturas, cortes de electricidad, etc. (Información obtenida directamente de los responsables de los datos de calidad del aire de la IMM)

IV. Diseño de base de datos

Luego de la limpieza y reconocimiento de los datos se pensaron 3 diseños posibles para la base de datos documental. Las consultas se realizarán sobre estos tres diseños y en base a ello se tratará de determinar el más adecuado para almacenar y manipular los datos de calidad del aire. En los tres diseños los metadatos correspondientes a las estaciones de la red, los contaminantes y los métodos de medición se incluyen en una colección aparte llamada `metadata`.

Toda la manipulación de datos se realizó en en Python[6]. La organización de los datos por contaminantes y por estaciones se realizó utilizando la biblioteca Pandas. La migración de los *DataFrame* de Pandas a formato JSON y posteriormente su migración a MongoDB se realizó utilizado la biblioteca PyMongo.

IV-A. Diseño 1

En primer lugar se realizó una migración directa de los documentos `.csv` a la base de datos documental. En esta base de datos cada colección contiene los datos anuales de todos los contaminantes y todas las estaciones de la red de monitoreo. El Listado 1 muestra un documento ejemplo de estas colecciones, tiene todo los campos que se encuentran en los archivos originales y se creó un índice en el campo `date`. En la Tabla II se muestran las colecciones en la base de datos `anuales_V1`.

Listado 1 Documento ejemplo - Diseño 1

```
{
  "_id": {
    "$oid": "648e0b79d1b42e9cf3f7aa6a"
  },
  "pollutant_id": "NO2",
  "pollutant_averaging": 1,
  "date": {
    "$date": "2022-08-23T16:00:00.000Z"
  },
  "pollutant_value": 14,
  "pollutant_unit": "ug/m3",
  "station_id": "UYMVD_E6",
  "X": 579229,
  "Y": 6142255,
  "ID_estacion": "Curva de Maronas",
  "method_id": "UYMVD_NO2_2"
}
```

IV-B. Diseño 2

Otra estrategia de diseño utilizada fue organizar las colecciones por contaminante. En este caso se organizaron todos los datos en 4 colecciones (correspondientes a los contaminantes considerados en este trabajo) y se definieron las colecciones como serie temporales, utilizando el formato de *time-serie* de MongoDB. El Listado 2 muestra un esquema ejemplo de los documentos de esta colección. El campo `timestamp` contiene la información temporal de la serie y de definió un

[6]El código completo se encuentra en el repositorio de GitLab

índice sobre él, en el objeto `metadata` se incluyen todos los demás campos relevantes y el campo `pollutant_value` contiene el valor de la medida del contaminante. En la Tabla II se muestran las colecciones en la base de datos `contaminantes_V2`.

Listado 2 Documento ejemplo - Diseño 2

```
{
  "timestamp":{
    "$date":"2016-01-01T00:00:00.000Z"
  },
  "metadata":{
    "method_id":"UYMVD_NO2_2",
    "pollutant_id":"NO2",
    "pollutant_unit":"ug/m3",
    "station_id":"UYMVD_E5"
  },
  "pollutant_value":14,
  "_id":{
    "$oid":"647933f3457f216e7a567841"
  }
}
```

IV-C. Diseño 3

Por último se organizaron las colecciones por punto de medición. En este caso se separó en 4 colecciones correspondientes a Ciudad Vieja (`UYMVD_E1`), Tres Cruces (`UYMVD_E5`), Curva de Maronas (`UYMVD_E6`) y Colón (`UYMVD_E8`). El nombre de cada colección corresponde al identificador asignado por la IMM a cada punto. Al igual que en el diseño 2 se definieron las colecciones como serie temporales, utilizando el formato de *time-serie* de MongoDB con la adición de la localización geográfica en un formato reconocido por MongoDB. El formato de geolocalización por los campos `is_location_exact`, `type` y `coordinates` especificando si es localización exacta, tipo de figura a dibujar en los mapas espaciales y coordenadas respectivamente. En la Tabla II se muestran las colecciones en la base de datos `estaciones_V3`.

Listado 3 Documento ejemplo - Diseño 3

```
{
  "_id":{
    "$oid":"6490adc044de57fc33035d80"
  },
  "metadata":{
    "pollutant_id":"PM2",
    "pollutant_unit":"ug/m3",
    "method_id":"UYMVD_PM2_b"
  },
  "timestamp":{
      "$date":"2016-01-01T00:00:00.000Z"
  },
  "pollutant_value":{
    "$numberDouble":"20.0"
  },
  "station_id":"UYMVD_E1",
  "location":{
    "is_location_exact":false,
    "type":"Point",
    "coordinates":[
      {
        "$numberDouble":"-56.20572796"
      },
      {
        "$numberDouble":"-34.906137122"
      }
    ]
  }
}
```

Tabla II: Diferentes colecciones implementadas en MongoDB

anuales_V1	contaminantes_V2	estaciones_V3
2016	NO2	UYMVD_E8
2017	PM2	UYMVD_E5
2018	PM10	UYMVD_E6
2019	O3	UYMVD_E1
2020	metadata	metadata
2021		
2022		
metadata		

Tabla III: Rangos de concentraciones de las categorías de calidad de aire

Categoría	PM2 ($\mu g/m^3$)	PM10 ($\mu g/m^3$)	NO2 ($\mu g/m^3$)	O3 ($\mu g/m^3$)
Muy buena	0-15	0-45	0-40	0-60
Buena	16-25	46-50	41-75	61-80
Moderada	26-35	51-75	76-200	81-100
Regular	36-75	76-150	201-500	101-160
Mala	>75	>150	>500	>160

V. Consultas sobre la base de datos

Luego del diseño de las colecciones y su migración a formato documental en MongoDB, se realizaron las mismas consultas sobre los tres diseños con el objetivo de determinar si alguno de ellos muestra ventajas respecto a los demás:

1. Determinar promedios mensuales históricos por contaminante
2. Determinar promedios mensuales históricos por contaminante para cada estación de la red
3. En base a los criterios de calidad[7] que se muestran en la Tabla III determinar el porcentaje de medidas en el año que los valores se encuentran en cada uno de los rangos establecidos.

El objetivo de estas consultas es realizar un análisis histórico de la calidad del aire en Montevideo y detectar si existen patrones anuales o mensuales observables.

La **consulta 1** consiste en determinar promedios mensuales históricos por contaminante, sin discriminar por estaciones. El Listado 4 muestra el código utilizado en el *pipeline* de agregación de MongoDB aplicado sobre el diseño 1 para determinar el promedio mensual histórico de $PM2$. Es necesario realizar varias operaciones de `$unionWith` para obtener el resultado final.

Listado 4 Consulta 1 sobre diseño 1 - Contaminante PM2

```
result = client['anuales_V1']['2016'].aggregate([
    {
        '$unionWith': {
            'coll': '2017'
        }
    }, {
        '$unionWith': {
            'coll': '2018'
```

[7]Rangos y valores definidos por la Red de Monitoreo de Calidad del Aire de la IMM. Detalle e informes disponibles la web.

```
        }
    }, {
        '$unionWith': {
            'coll': '2019'
        }
    }, {
        '$unionWith': {
            'coll': '2020'
        }
    }, {
        '$unionWith': {
            'coll': '2021'
        }
    }, {
        '$unionWith': {
            'coll': '2022'
        }
    }, {
        '$match': {
            'pollutant_id': 'PM2'
        }
    }, {
        '$group': {
            '_id': {
                '$month': '$date'
            },
            'prom_mensual': {
                '$avg': '$pollutant_value'
            }
        }
    }, {
        '$project': {
            '_id': 0,
            'month': '$_id',
            'prom_mensual': 1
        }
    }, {
        '$sort': {
            'month': 1
        }
    }
])
```

La consulta 1 sobre el diseño 2 para el contaminante $PM2$ es mostrada en el Listado 5. Debido a la organización de las colecciones por contaminantes, no es necesario realizar búsquedas sobre otras colecciones, lo que simplifica la consulta respecto a los demás diseños.

Listado 5 Consulta 1 sobre diseño 2 - Contaminante PM2

```
result = client['contaminantes_V2']['PM2'].aggregate([
    {
        '$group': {
            '_id': {
                '$month': '$timestamp'
            },
            'prom_mensual': {
                '$avg': '$pollutant_value'
            }
        }
    }, {
        '$project': {
            '_id': 0,
            'month': '$_id',
            'prom_mensual': 1
        }
    }
])
```

La misma consulta aplicada sobre el diseño 3 se muestra en el Listado 6. Nuevamente es necesario realizar búsquedas sobre varias colecciones para obtener el promedio histórico mensual de un contaminante.

Listado 6 Consulta 1 sobre diseño 3 - Contaminante PM2

```
result = client['estaciones_V3']['UYMVD_E1'].aggregate([
    {
        '$unionWith': {
            'coll': 'UYMVD_E5'
        }
    }, {
        '$unionWith': {
            'coll': 'UYMVD_E6'
        }
    }, {
        '$unionWith': {
            'coll': 'UYMVD_E8'
        }
    }, {
        '$match': {
            'metadata.pollutant_id': 'PM2'
        }
    }, {
        '$group': {
            '_id': {
                '$month': '$timestamp'
            },
            'prom_mensual': {
                '$avg': '$pollutant_value'
            }
        }
    }, {
        '$project': {
            '_id': 0,
            'month': '$_id',
            'prom_mensual': 1
        }
    }
    , {
        '$sort': {
            'month': 1,
        }
    }
])
```

Tabla IV: Promedio mensual histórico de PM2 - Consulta 1

mes	prom	mes	prom
1	16,48	7	32,35
2	19,45	8	41,30
3	24,40	9	38,16
4	30,43	10	37,22
5	26,03	11	33,05
6	22,43	12	20,55

El resultado obtenido para la consulta 1 es idéntico para los tres diseños, se muestra en la Tabla IV para el contaminante $PM2$. La consulta para los demás contaminantes es análoga y se puede encontrar en el repositorio de GitLab.

La **consulta 2** consiste en determinar los promedio históricos mensuales por estación, para cada contaminante. Para realizar la consulta sobre el diseño 1 es necesario nuevamente realizar las operaciones de `$unionWith` sobre las demás colecciones anuales, y sobre el diseño 3 requiere las operaciones de `$unionWith` sobre las demás colecciones de estaciones; luego de la unión de colecciones para el diseño 1 y 3 es necesario realizar un `$match` para el contaminante deseado. Luego de estas primeras operaciones, las operaciones de agregación son las mismas en las tres colecciones y se muestran en el Listado 7 para el contaminante $PM2$. En el diseño 2 la consulta se puede realizar directamente sobre la colección del contaminante deseado. El resultado de la consulta se muestra en la Figura 3 para el contaminante $PM2$.

Listado 7 Consulta 2 sobre diseño 2 - Contaminante PM2

```
result = client['contaminantes_V2']['NO2'].aggregate([
    {
```

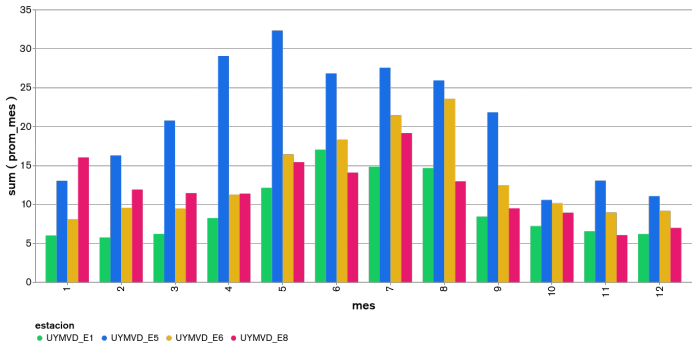

Figura 3: Promedio mensual histórico de PM2 por estación (visualización consulta 2).

```
        '$group': {
            '_id': {
                'mes': {
                    '$month': '$timestamp'
                },
                'estacion': '$metadata.station_id'
            },
            'prom_mes': {
                '$avg': '$pollutant_value'
            }
        }
    }, {
        '$project': {
            '_id': 0,
            'mes': '$_id.mes',
            'estacion': '$_id.estacion',
            'prom_mes': '$prom_mes'
        }
    }, {
        '$sort': {
            'mes': 1,
            'estacion': 1
        }
    }
])
```

La **consulta 3** busca determinar la cantidad medidas en el año que los valores se encuentran en cada uno de los rangos de calidad establecidos en la Tabla III. La consulta se realizó sobre los tres diseños, al igual que la consulta 2, los diseños 1 y 3 requieren previamente realizar `unionWith` y `match` sobre las diferentes colecciones mientras que diseño 2 permite realizar la consulta directamente sobre la colección del contaminante deseado. Luego de unir las diferentes colecciones, la consulta es similar en los tres diseños, en el Listado 8 se presenta a modo de ejemplo la consulta sobre el diseño 2 para el contaminante $PM2$.

Listado 8 Consulta 3 sobre diseño 2 - Contaminante PM2

```
result = client['contaminantes_V2']['PM2'].aggregate([
    {
        '$addFields': {
            'classification': {
                '$switch': {
                    'branches': [
                        {
                            'case': {
                                '$lte': [
                                    '$pollutant_value', 15
                                ]
                            },
                            'then': 'Muy Bueno'
                        }, {
                            'case': {
                                '$lte': [
                                    '$pollutant_value', 25
                                ]
```

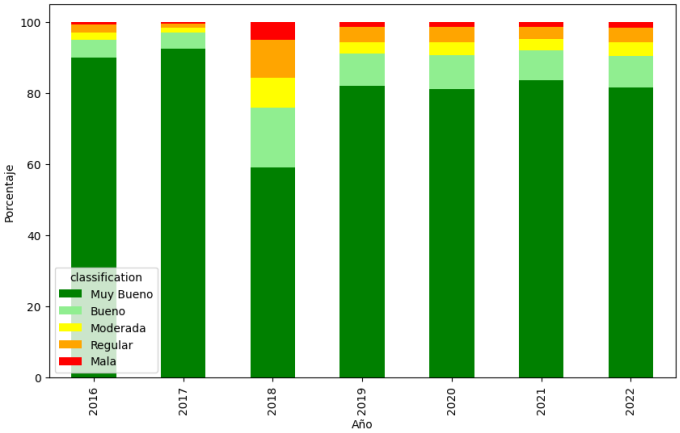

Figura 4: Porcentaje de medidas en cada rango de calidad por año (visualización consulta 3).

```
                            },
                            'then': 'Bueno'
                        }, {
                            'case': {
                                '$lte': [
                                    '$pollutant_value', 35
                                ]
                            },
                            'then': 'Moderada'
                        }, {
                            'case': {
                                '$lte': [
                                    '$pollutant_value', 75
                                ]
                            },
                            'then': 'Regular'
                        }
                    ],
                    'default': 'Mala'
                }
            }
        }
    }, {
        '$group': {
            '_id': {
                'year': {
                    '$year': '$timestamp'
                },
                'classification': '$classification'
            },
            'count': {
                '$sum': 1
            }
        }
    }, {
        '$project': {
            '_id': 0,
            'year': '$_id.year',
            'classification': '$_id.classification',
            'count': 1
        }
    },{
        '$sort' : {
            'year': 1
        }
    }
])
```

Los resultados obtenidos en la consulta 3 sobre los tres diseños son idénticos, en la Figura 4 se presenta una visualización del resultado obtenido.

VI. Conclusiones y trabajo futuro

Los trabajos relacionados analizados muestran que, si bien MongoDB es una solución adecuada para trabajar con series

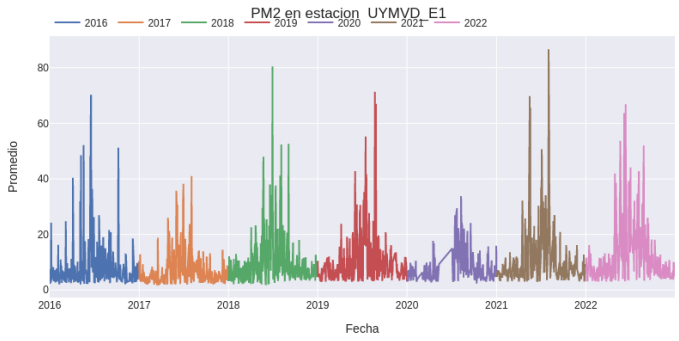

Figura 5: Valores de PM2 en el período 2016-2022

temporales, no siempre tiene el mejor rendimiento, en particular cuando se compara con motores relacionales, específicamente PostgreSQL e InfluxDB. De todas formas, las características y herramientas que ofrece MongoDB, en particular MongoDB Atlas (como los charts y la definición nativa de series temporales) hace que sea una solución interesante al momento de trabajar con datos de sensores.

Para comparar los tres diseños implementados y decidir cuál fue el mejor, se tomó el criterio de simplicidad de las consultas, sin tomar en cuenta aspectos cuantitativos de rendimiento. De los tres diseños analizados en este trabajo, podemos decir que el que tuvo mejores resultados fue el diseño 2, dónde los datos se organizaron por contaminante. Las tres consultas pueden realizarse con pocas agregaciones en MongoDB. La organización por contaminantes favorece el análisis histórico y búsqueda de tendencias que se planteó en este trabajo.

El diseño 2 permite realizar visualizaciones de manera sencilla, con el objetivo de observar el comportamiento de cada contaminante; a modo de ejemplo la Figura 5 se muestran los valores históricos de $PM2$ en la estación UYMVD_E1, es posible observar que existe una tendencia que se repite año a año, con períodos en los cuales el valor del contaminante es más elevado.

Para profundizar en el uso de la herramienta MongoDB Charts se realizó un panel de visualización (*dashboard*) de Calidad del aire en Montevideo que sistematiza los principales resultados obtenidos en el análisis de los datos sobre el diseño documental.

Para mejorar el alcance y los resultados de esta investigación se sugieren algunas líneas de trabajo a futuro:

- Realizar una comparación cuantitativa sobre el desempeño de los diseños implementados, determinando el *workload* sobre cada diseño.
- Aplicar técnicas de aprendizaje automático para la imputación de los valores `NA` y *forecasting* para predicción del comportamiento de los contaminantes.
- Comparar los diseños implementados en MongoDB con otro motor de base de datos no relacional.

Agradecimientos

Agradecemos al Ing. Quím. Pablo Franco de la Unidad Calidad de Aire (IMM) por los datos brindados, la buena disposición y el tiempo dedicado a aclarar nuestras dudas sobre los mismos.

Referencias

Di Martino, S., Fiadone, L., Peron, A., Riccabone, A., and Vitale, V. N. (2019). Industrial internet of things: persistence for time series with nosql databases. In *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 340–345. IEEE.

Eyada, M. M., Saber, W., El Genidy, M. M., and Amer, F. (2020). Performance evaluation of iot data management using mongodb versus mysql databases in different cloud environments. *IEEE access*, 8:110656–110668.

Harrison, G. (2015). *Next Generation Databases: NoSQLand Big Data*. Apress.

Makris, A., Tserpes, K., Spiliopoulos, G., and Anagnostopoulos, D. (2019). Performance evaluation of mongodb and postgresql for spatio-temporal data. In *EDBT/ICDT Workshops*.

Mehmood, N. Q., Culmone, R., and Mostarda, L. (2017). Modeling temporal aspects of sensor data for mongodb nosql database. *Journal of Big Data*, 4(1):8.

MongoDB (2019). *Time Series Data and MongoDB: Best Practices Guide*. A MongoDB White Paper.

MongoDB (2023). Internet of things (iot) databases. https://www.mongodb.com. Accedido: 4 de junio, 2023.

Nielsen, A. (2019). *Practical time series analysis: Prediction with statistics and machine learning*. O'Reilly Media.

Ramesh, D., Sinha, A., and Singh, S. (2016). Data modelling for discrete time series data using cassandra and mongodb. In *2016 3rd international conference on recent advances in information technology (RAIT)*, pages 598–601. IEEE.

Widyani, Y., Laksmiwati, H., and Bangun, E. D. (2016). Mapping spatio-temporal disaster data into mongodb. In *2016 International Conference on Data and Software Engineering (ICoDSE)*, pages 1–5. IEEE.