

# Fundamentos bases de datos NoSQL: MongoDB

Elena Alcalá Contreras

*m.elenaalcala@gmail.com*

Engineering Lead - Digital Products  
JLL/T

Marzo/Abril 2022

# ÍNDICE

1. NoSQL: Conceptos clave
2. MongoDB: Conceptos y manejo
3. ATLAS Cloud: Práctica
4. Consultas en MongoDB
5. Ejercicios prácticos
6. MongoDB desde Python
7. Referencias

# NoSQL

## Conceptos clave

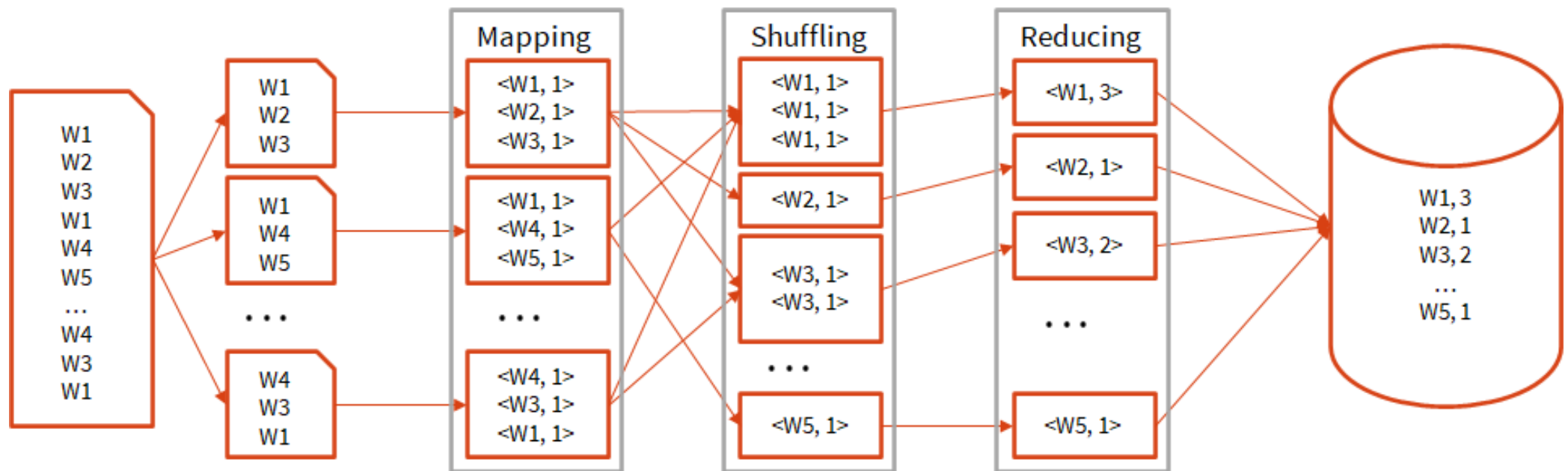
# ¿Por qué NoSQL?

- Estructuras no tabulares
  - Pares clave-valor
  - Columnas
  - **Documentos**
  - Grafos
- Consulta y modificación no por SQL
- Escalabilidad horizontal
- Flexibilidad
- No necesidad de escalar verticalmente (hardware)

# Paradigma MapReduce

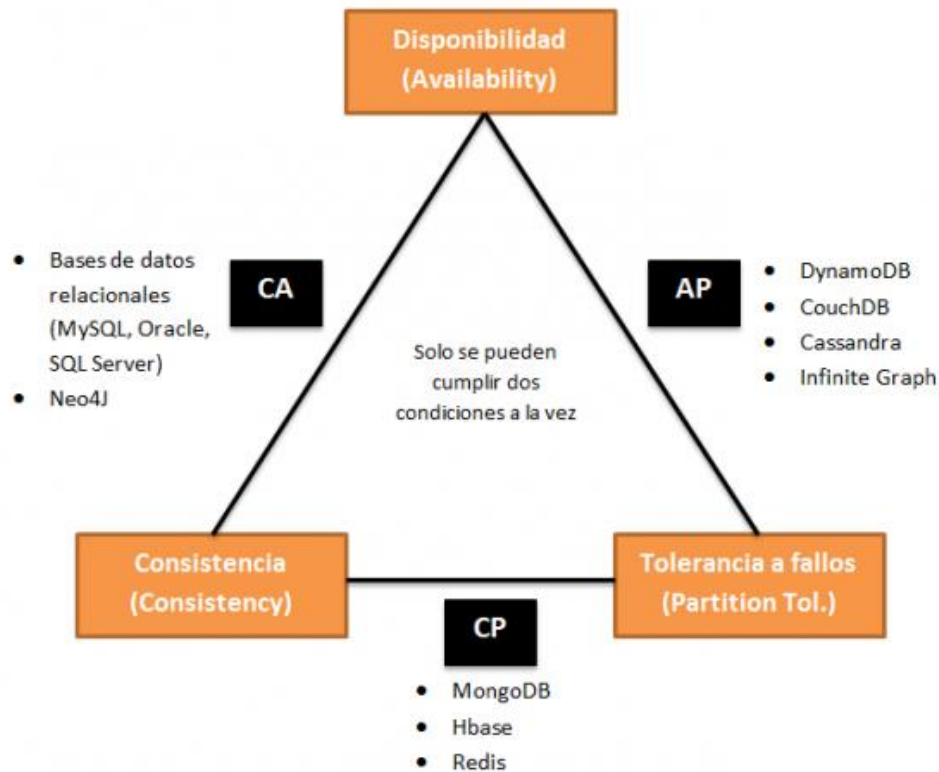
Modelo de programación para dar soporte a la computación paralela sobre grandes colecciones de datos

- **Map** convertir fichero de gran tamaño en secuencia de pares <clave, valor>
- **Suffle** recolectar y ordenar según la clave
- **Reduce** combinar los valores asociados a cada clave según el problema



# Teorema CAP

- **Consistency** todos los nodos ven la misma información al mismo tiempo
- **Availability** peticiones a un nodo disponible deben obtener respuesta razonable
- **Partition Tolerance** el sistema sigue funcionando aunque algunos nodos fallen



**Imposible garantizar estas tres características de forma simultánea en bases de datos no relacionales**

# Modelo ACID vs BASE

## ACID

- **Atomic** Transacciones completas
- **Consistent** Datos intactos
- **Isolated** Operaciones aisladas
- **Durable** Persistencia de la transacción

Típico de bbdd relacionales

## BASE

- **Basic Availability** Respuesta a cualquier solicitud
- **Soft-state** Mismos valor para un elemento si no se realizan actualizaciones
- **Eventual consistency** Propagación de actualizaciones

Típico de bbdd no relacionales

# Bases de datos documentales

- Tipo de bases de datos NoSQL en la que los datos están organizados en forma de documentos
- Cada base de datos documental emplea un estándar específico para representar los documentos: XML, **JSON/BSON**, YAML
- Tipología de uso basada en la realización de muchas **inserciones y consultas** pero muy pocas (o inexistentes) actualizaciones



## Tienen **estructura**

Conjunto de campos y valores



## **Campos multivalor**

Un campo puede tener asignada una lista de valores



## **No tienen esquema (schema-less)**

Cada documento puede tener un conjunto distinto de campos



## **Jerarquía: documentos embebidos**

Un documento puede estar asignado como valor



# JSON

- Formato de texto plano ligero para intercambio de datos
- Costoso de almacenar. Usar **BSON** para almacenamiento eficiente
- Leído por cualquier lenguaje de programación
- Alternativa a XML
- Tipos aceptados: número (entero/float), string (comillas dobles), booleano (true/false), array ([]), objeto ({})

```
{  
  "id": 1000,  
  "city": "Madrid",  
  "isCapital": true,  
  "towns": [  
    {  
      "id": 28,  
      "name": "Madrid"  
    },  
    {  
      "id": 172,  
      "name": "Las Rozas de Madrid"  
    }  
  ],  
  "hasSea": null,  
  "pibVar": -11.1  
}
```



# MongoDB

## Conceptos y manejo

# ¿Qué es MongoDB?

- Proviene de la palabra *humongous*
- Es una **base de datos NoSQL documental**
- **Código abierto**
- **No soporta SQL**
- Utiliza **BSON** para representar la información internamente
- Acepta documentos **JSON** como fuente de información
- Sigue el **modelo CP** según el teorema CAP

# Breve Historia



# Ventajas y Desventajas

## VENTAJAS

- **Schema-less**
- 1 objeto = 1 documento = **CLARIDAD**
- **SHARDING** y **REPLICATION**
- Consulta potente por **ÍNDICES**
- Soporte **GEO** avanzando
- Documentación y herramientas
- Integración en lenguajes modernos

## DESVENTAJAS

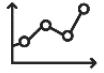
- No es sencillo realizar **JOIN** entre colecciones
- Limitación en niveles de documentos embebidos
- Limitación en tamaño de documentos a **16MB**
- No garantiza **ACID**
- No recomendado para casos con muchas **actualizaciones**

# ¿Cuándo usar MongoDB?

- Performance elevada para **búsquedas**
- Sistemas con **recursos geográficos**
- No es tan relevante garantizar ACID
- Escenarios de BigData/DataHub
- **Cacheo**

**MongoDB es válido como sistema central de datos pero es el complemento perfecto a un RDBMS**

# Ejemplos reales de aplicación



MapReduce para análisis de datos



Repositorio testigos de mercado



Log de aplicación



Auditoría



Gestión de comentarios en redes



Información geográfica y mapas



Configuración de aplicación



Motores de cálculo

# Elementos de MongoDB

- **Servidor**

Instancia de MongoDB

- **Base de datos**

Contenedor de colecciones

- **Colección**

Agrupación de documentos similares o relacionados

Cada colección pertenece a una única base de datos

- **Documento**

Conjunto de campos clave-valor

Sin esquema fijo

Un documento pertenece a una única colección

- **ObjectId (\_id)**

Identificador único

Creado, gestionado y asignado automáticamente

**No existe el concepto de FOREIGN KEY. No existe el concepto de JOIN**



# Documentos embebidos vs Referencias

## Documentos embebidos

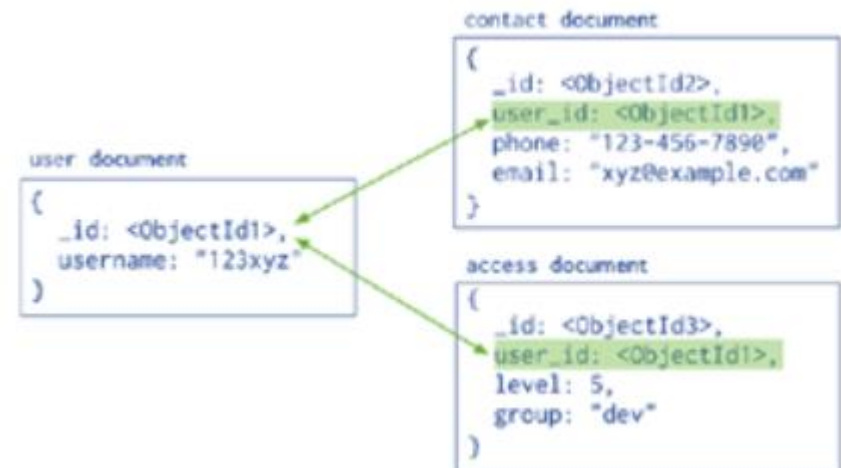
- Minimiza el número de consultas
- No actualización de la información embebida
- Consulta simultánea con el documento principal
- Número razonable de niveles



- Redundancia y mayor consumo de disco

## Referencias

- Múltiples consultas
- Información relacionada que sufre cambios frecuentes
- Información relacionada consumida de forma independiente
- Varios niveles de relación



- Poca redundancia y menor consumo

# MongoDB vs RDBMS

MongoDB	RDBMS
Base de datos	Base de datos
Colección	Tabla, Vista
Documento	Fila
Campo	Columna
ObjectId (_id)	Clave primaria
Referencia	Clave foránea
Documento embebido	Join

# Conceptos avanzados

- **Réplica**

Mongos con la **misma información**. En MongoDB existe la auto-réplica, que mejora la redundancia y la capacidad de lectura

- **Sharding**

Estrategia o método para **distribuir datos** en máquinas. Solo en casos de muchos datos. Consultas distribuidas. Complica la arquitectura

- **Modelización**

El modelo de datos viene definido por el uso. Colecciones lo más homogéneas posible. **ÍNDICES**

- **Interacción**

Tres formas de interacción:

- **Consola** cliente de MongoDB – menos útil
- **Aplicación** cliente de MongoDB (Studio 3T, Robo 3T, Compass) – gestión cómoda
- Conexión desde **lenguajes** de programación (Python, R, C#) – disposición de drivers

# ATLAS Cloud Práctica

# ATLAS Cloud

- **DBaaS** Servicio de base de datos Mongo como servicio
- Servicio de “0 mantenimiento”
- Servicio de **pago por uso**

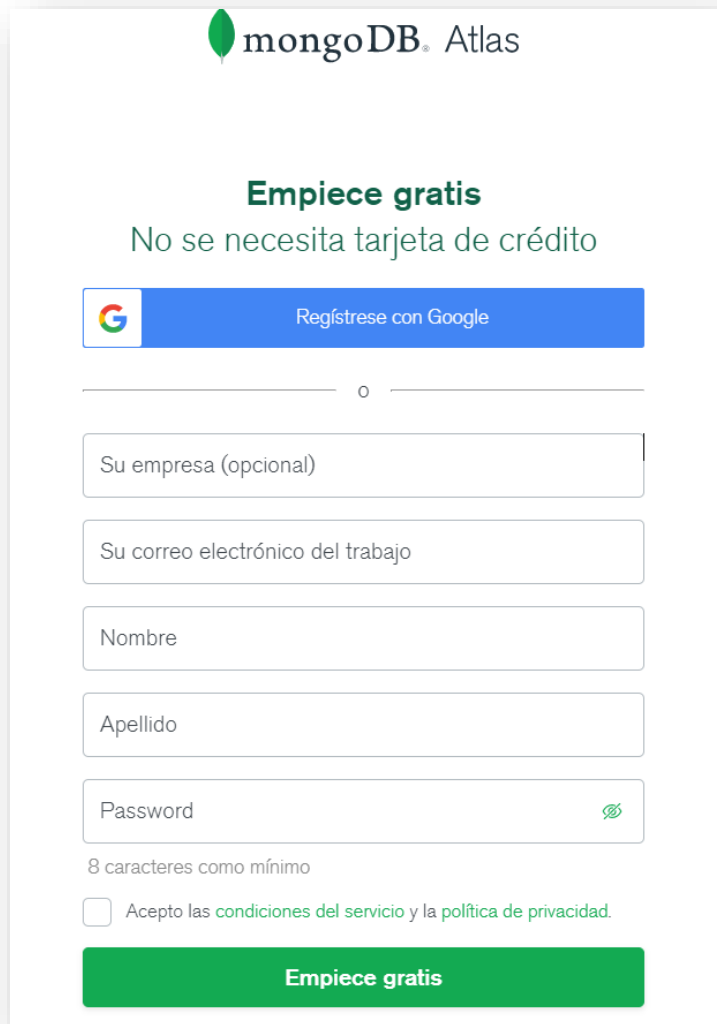
## VENTAJAS

- A la larga, menor coste
- No CAPEX (bienes físicos) pero sí **OPEX** (operaciones y servicios)
- Mejor mantenimiento posible

## DESVENTAJAS

- Alto coste para poco uso
- Posibles problemas para requerimientos particulares de seguridad

# ATLAS Cloud | Alta en el servicio



The registration form for MongoDB Atlas is displayed. It features the MongoDB Atlas logo at the top. Below the logo, the text "Empiece gratis" (Get started for free) is shown in green, followed by "No se necesita tarjeta de crédito" (No credit card needed). A blue button with the Google logo and the text "Regístrese con Google" (Sign up with Google) is present. Below this, a horizontal line with a small circle in the center separates the Google sign-up option from the manual registration fields. The manual registration fields include: "Su empresa (opcional)" (Your company (optional)), "Su correo electrónico del trabajo" (Your work email), "Nombre" (First name), "Apellido" (Last name), and "Password" (with a strength indicator icon). Below the password field, it says "8 caracteres como mínimo" (At least 8 characters). A checkbox for "Acepto las condiciones del servicio y la política de privacidad" (I accept the terms of service and privacy policy) is located below the password field. At the bottom, a green button labeled "Empiece gratis" (Get started for free) is visible.

## Pasos a seguir:



The "Connect to Atlas" checklist is shown. It has a title "Connect to Atlas" and a subtitle "Follow this checklist to get started." Below the subtitle, a progress bar shows "0%". The checklist items are: "Build your first cluster", "Create your first database user", "Whitelist your IP address", and "Connect to your cluster". Each item is preceded by an unchecked radio button. At the bottom right, there is a link that says "No Thanks!".

<https://www.mongodb.com/es/cloud/atlas/register>

# ATLAS Cloud | Creación cluster



## Create a database

Choose your cloud provider, region, and specs.

Build a Database

FREE

Shared

For learning and exploring MongoDB in a cloud environment. Basic configuration options.

- ✓ No credit card required to start
- ✓ Explore with sample datasets
- ✓ Upgrade to dedicated clusters for full functionality

Create

Starting at  
**FREE**

PREVIEW Serverless

Dedicated

FREE Shared

Cloud Provider & Region

AWS, Ireland (eu-west-1) ^

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage)  
Encrypted ^

Additional Settings

MongoDB 5.0, No Backup ^

Cluster Name

Afi2022 ^

**FREE**

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

[Back](#)

Create Cluster

# ATLAS Cloud | Alta usuarios

## 1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Username and Password

Certificate

Create a database user using a username and password. Users will be given the *read and write to any database* [privilege](#) by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username

afidbuser2022

Password 

.....

 Autogenerate Secure Password

 Copy

Create User



# ATLAS Cloud | Seguridad

## 2 Where would you like to connect from?

Enable access for any network(s) that need to read and write data to your cluster.

ADVANCED

### My Local Environment

Use this to add network IP addresses to the IP Access List. This can be modified at any time.

### Cloud Environment

Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

### Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters.

IP Address

Description

Enter IP Address

Enter description

Add Entry

Add My Current IP Address

# ATLAS Cloud | Conexión

## Database Deployments

Find a database deployment...

Afi2022

Connect

View Monitoring

Browse Collections

...

R 0

W 0

Last 6 minutes

100.0/s

i

Connections 0

Last 6 minutes

100.0

i

In 0.0 B/s

Out 0.0 B/s

Last 6 minutes

100.0 B/s

VERSION

5.0.6

REGION

AWS / Ireland (eu-west-1)

CLUSTER TIER

M0 Sandbox (General)

TYPE

Replica Set - 3 nodes

BACKUPS

Inactive

I do not have MongoDB Compass

I have MongoDB Compass

1 Choose your version of Compass:

1.12 or later

See your Compass version in "About Compass"

2 Copy the connection string, then open MongoDB Compass.

mongodb+srv://afidbuser2022:<password>@afi2022.hpotk.mongodb.net/test

You will be prompted for the password for the **afidbuser2022** user's (Database User) username. When entering your password, make sure that any special characters are [URL encoded](#).



### Connect with the MongoDB Shell

Interact with your cluster using MongoDB's interactive Javascript interface



### Connect your application

Connect your application to your cluster using MongoDB's native drivers

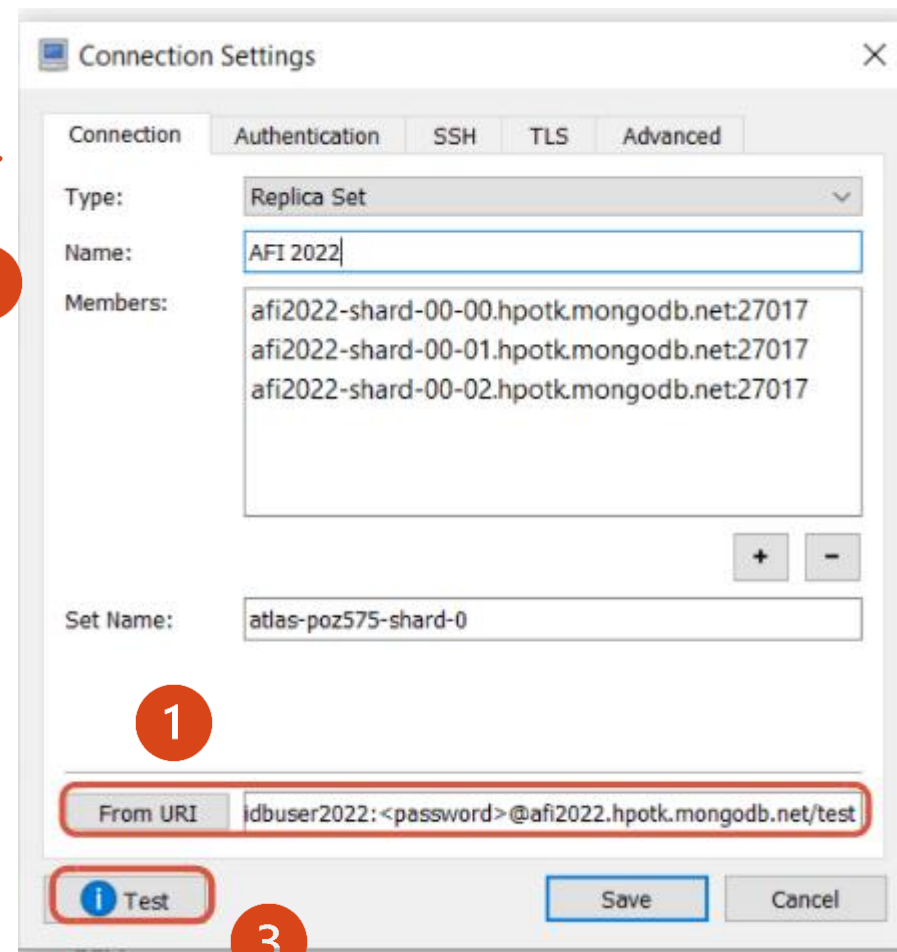
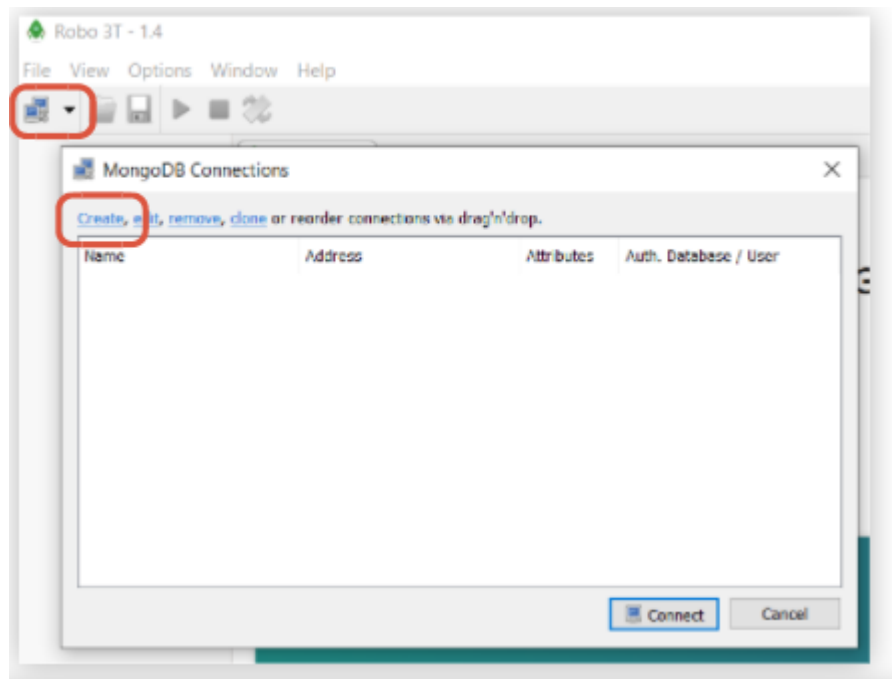


### Connect using MongoDB Compass

Explore, modify, and visualize your data with MongoDB's GUI

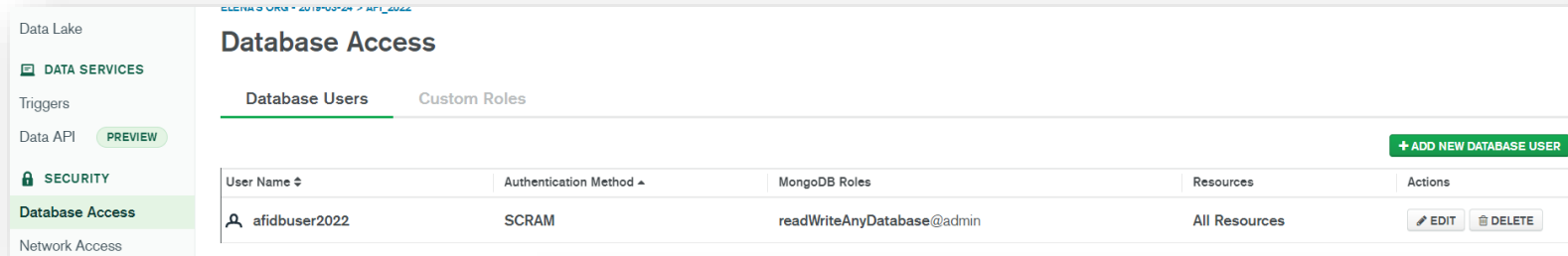


# Robo3T | Conexión al cluster



Sustituir “<password>” por la contraseña generada

# ATLAS Cloud | Modificar permisos del usuario



Data Lake

DATA SERVICES

Triggers

Data API PREVIEW

SECURITY

Database Access

Network Access

## Database Access

Database Users Custom Roles

+ ADD NEW DATABASE USER

User Name	Authentication Method	MongoDB Roles	Resources	Actions
afidbuser2022	SCRAM	readWriteAnyDatabase@admin	All Resources	<span>EDIT</span> <span>DELETE</span>

## Database User Privileges

Configure role based access control by assigning database users a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. **You must choose at least one role or privilege.** [Learn more about roles.](#)

### Built-in Role

1 SELECTED

Select one [built-in role](#) for this user.

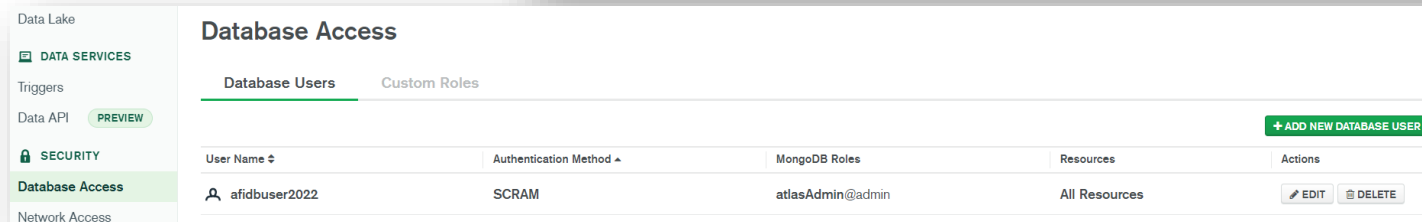
Read and write to any database

Atlas admin

✓ Read and write to any database

Only read any database

custom role in the [Custom Roles](#) tab.



Data Lake

DATA SERVICES

Triggers

Data API PREVIEW

SECURITY

Database Access

Network Access

## Database Access

Database Users Custom Roles

+ ADD NEW DATABASE USER

User Name	Authentication Method	MongoDB Roles	Resources	Actions
afidbuser2022	SCRAM	atlasAdmin@admin	All Resources	<span>EDIT</span> <span>DELETE</span>

# Consultas en MongoDB

# Interactuar con los datos

- **Crear colección**

**`db.createCollection(<name>, <options>)`**

<name>: nombre de la colección que se va a crear

<options>: documento (opcional) para especificar opciones de la colección como restricciones de contenido, tamaño máximo de la colección o número máximo de documentos de la colección

- **Eliminar colección**

**`db.collection.drop()`**

- **Eliminar base de datos**

**`db.dropDatabase()`**



Las operaciones de eliminar, borran la colección o base de datos junto con todo su contenido y **NO SE PUEDEN DESHACER**

<https://docs.mongodb.com/>

# Interactuar con los datos | CRUD

- **Create**

- `insertOne(data, options)`
  - `insertMany(data, options)`

- **Read**

- `find(filter, options)`
  - `findOne(filter, options)`

- **Update**

- `updateOne(filter, data, options)`
  - `updateMany(filter, data, options)`
  - `replaceOne(filter, data, options)`

- **Delete**

- `deleteOne(filter, options)`
  - `deleteMany(filter, options)`

# CRUD | Inserción de documentos

```
db.collection.insertOne(<document>)
```

```
db.collection.insertMany( [ <document>, <document>, ... ] )
```

*[Mongo] db.collection.insertOne({ Field1: 'Value1', Field2: Value2 })*

*[SQL] INSERT INTO collection(Field1, Field2) VALUES ('Value1', Value2)*

Al insertar varios documentos con la instrucción `insertMany()`, si alguno genera conflicto y no se puede insertar, automáticamente se PARA la inserción. Los documentos previos sí habrán sido insertados pero no el mencionado ni los siguientes. Para evitarlo, se puede emplear el comando:

```
db.collection.insertMany([<document>, <document>, ...], {ordered: false})
```

que permite realizar aquellas inserciones que sean posibles y mostrar un error final indicando qué documentos no se han podido insertar en la colección



# CRUD | Inserción de documentos | Ejemplo

1. Crear una colección con el nombre **TEST\_COLLECTION**
2. Añadir un documento con la siguiente información:

```
{  
  "item": "AFI",  
  "tags": ["white", "orange"],  
  "qty": 145,  
  "size": {  
    "height": 1500,  
    "width": 400  
  }  
}
```

3. Añadir dos documentos con la misma estructura que el anterior

# CRUD | Inserción de documentos | Ejemplo

```
# Create collection
db.createCollection("TEST_COLLECTION")

# Add one document
db.TEST_COLLECTION.insertOne({
  "item": "AFI",
  "tags": ["white", "orange"],
  "qty": 145,
  "size": {
    "height": 1500,
    "width": 400
  }
})
```

```
# Add multiple documents
db.TEST_COLLECTION.insertMany([
  {
    "item": "JLL",
    "tags": ["black", "white", "red"],
    "qty": 530,
    "size": {
      "height": 7800,
      "width": 320
    }
  }, {
    "item": "EXAMPLE",
    "tags": ["yellow"],
    "qty": 40,
    "size": {
      "height": 300,
      "width": 64
    }
  }
])
```

# cRUD | Recuperación de documentos

**`db.collection.find(<filter>)`**

<filter>: representa un documento con definición de filtros a aplicar. Si no se quiere aplicar ningún filtro, se puede omitir el parámetro o emplear el documento vacío {}

*[Mongo] db.collection.find()*

*[SQL] SELECT \* FROM collection*

*[Mongo] db.collection.find({ Field1: 'Value1' })*

*[SQL] SELECT \* FROM collection WHERE Field1='Value1'*

# CRUD | Recuperación de documentos

## Comparación

Filtro	Sintaxis
Igualdad	{<key>: <value>} / {<key>: {\$eq: <value>}}
Desigualdad	{<key>: {\$neq: <value>}}
Menor o igual que	{<key>: {\$lte: <value>}}
Mayor o igual que	{<key>: {\$gte: <value>}}
Contenido en	{<key>: {\$in: [<value>, ...]}}
No contenido en	{<key>: {\$nin: [<value>, ...]}}

[Mongo] `db.collection.find({ Field2: {$gt: Value2 } })`

[SQL] `SELECT * FROM collection WHERE Field2 > Value2`

## Estructura

Filtro	Sintaxis
Existencia de campo	{<key>: {\$exists: <bool>}}
Tipo de campo	{<key>: {\$type: <mongotype>}} / {<key>: {\$type: [<mongotype>, ...]}}

# cRUD | Recuperación de documentos

## Lógicos

Filtro	Sintaxis
AND	<code>{ \$and: [&lt;filter&gt;, ...] }</code>
OR	<code>{ \$or: [&lt;filter&gt;, ...] }</code>
NOR	<code>{ \$nor: [&lt;filter&gt;, ...] }</code>
NOT	<code>{ \$not: &lt;filter&gt; }</code>

[Mongo] `db.collection.find({ $and: [ { Field1: "Value1" }, { Field2: { $lt: Value2 } } ] })`

[SQL] `SELECT * FROM collection WHERE Field1 = 'Value1' and Field2 < Value2`

## Evaluación

Filtro	Sintaxis
Comprobación de esquema	<code>{ \$jsonSchema: &lt;document&gt; }</code>
Expresiones regulares	<code>{ &lt;key&gt;: { \$regex: &lt;expression&gt; } }</code>
Comparación de campos	<code>{ \$expr: &lt;filter&gt; }</code>

# cRUD | Recuperación de documentos

## Anidados

Filtro	Sintaxis
Documento completo	{<key>: <document>}
Campo anidado	{"<key>.<subkey>": <filter>}

[Mongo] `db.collection.find({ "Field1.SubField": "ValueNested" })`

## Arrays

Filtro	Sintaxis
Array parcial	{<key>: <value>}
Array completo (con orden)	{<key>: [<value>, ...]}
Array completo (sin orden)	{<key>: {\$all: [<value>, ...]}}
Número de elementos	{<key>: {\$size: <num>}}
Comprobar condiciones	{<key>: {\$elemMatch: <document>}}

[Mongo] `db.collection.find({ ArrayField: { $size: 6 } })`

[Mongo] `db.collection.find({ ArrayField: { $elemMatch: { $gt: 4 } } })`

# cRUD | Recuperación de documentos | Ejemplo

1. Recuperar aquellos documentos cuyo campo **item** sea igual a “AFI”
2. Recuperar aquellos documentos cuyo campo **qty** sea mayor que 100
3. Recuperar aquellos documentos cuyo campo **width** sea menor que 300
4. Recuperar aquellos documentos que contengan el **tag** “white”
5. Recuperar aquellos documentos que tengan únicamente dos **tags**
6. Recuperar aquellos documentos cuyo campo **height** sea menor o igual a 8000 y cuyo campo **qty** sea mayor que 500

# cRUD | Recuperación de documentos | Ejemplo

*# Campo item igual a AFI*

```
db.TEST_COLLECTION.find({ item: "AFI" })
```

*# Campo qty > 100*

```
db.TEST_COLLECTION.find({ qty: { $gt: 100 } })
```

*# Campo width < 300*

```
db.TEST_COLLECTION.find({ "size.width": { $lt: 300 } })
```

*# Campo tags contenga white*

```
db.TEST_COLLECTION.find({ tags: { $elemMatch: { $eq: "white" } } })
```

*# Documentos con solo dos tags*

```
db.TEST_COLLECTION.find({ tags: { $size: 2 } })
```

*# Campo height <= 8000 y qty > 500*

```
db.TEST_COLLECTION.find({ $and: [{ "size.height": { $lte: 8000 } }, { qty: { $gt: 500 } } ] })
```



# cRUD | Contar documentos y seleccionar campos

- Contar resultados de una consulta

`db.collection.count()`

`db.collection.count(<filter>)`

`db.collection.find(<filter>).count()`

*[Mongo] db.collection.count({ Field1: "Value1" })*

*[SQL] SELECT COUNT(\*) FROM collection WHERE Field1 = 'Value1'*

- Seleccionar campos a mostrar en los documentos devueltos

`db.collection.find(<filter>, <projection>)`

<filter>: documento con los filtros a aplicar – si es necesario -

<projection>: documento que contiene como claves los nombres de los campos existentes en el documento y un flag 0/1 indicando si el campo debe ser devuelto o no

**Por defecto, el campo `_id` siempre se devuelve a no ser que se indique explícitamente**

*[Mongo] db.collection.find({}, { Field1: 1, Field2: 0 })*

*[SQL] SELECT Field1 FROM collection*

# cRUD | Limitar y ordenar resultados

- Limitación sobre el número de documentos devueltos

**`db.collection.find(<filter>).limit(<limit>).skip(<skip>)`**

<limit>: número máximo de resultados a devolver

<skip>: número de documentos que se quieren saltar

*[Mongo] db.collection.find().limit(10).skip(5)*

*[SQL] SELECT \* FROM collection OFFSET 5 ROWS FETCH NEXT 10 ROWS ONLY*

- Ordenación sobre documentos devueltos

**`db.collection.find(<filter>).sort(<criteria>)`**

<criteria>: documento que contiene como clave los nombre de los campos por los que se quiere ordenar y, como valor, un flag 1/-1 que indica si la ordenación es ascendente o descendente

*[Mongo] db.collection.find({ Field1: "Value1" }).sort({ Field1: -1 })*

*[SQL] SELECT \* FROM collection WHERE Field1 = 'Value1' ORDER BY Field1 DESC*

# CRUD | Valores no duplicados

**`db.collection.distinct(<field>, <query>)`**

<field>: campo para el que se quieren obtener los valores sin duplicados

<query>: query opcional para especificar el filtro previo a la selección de valores no duplicados

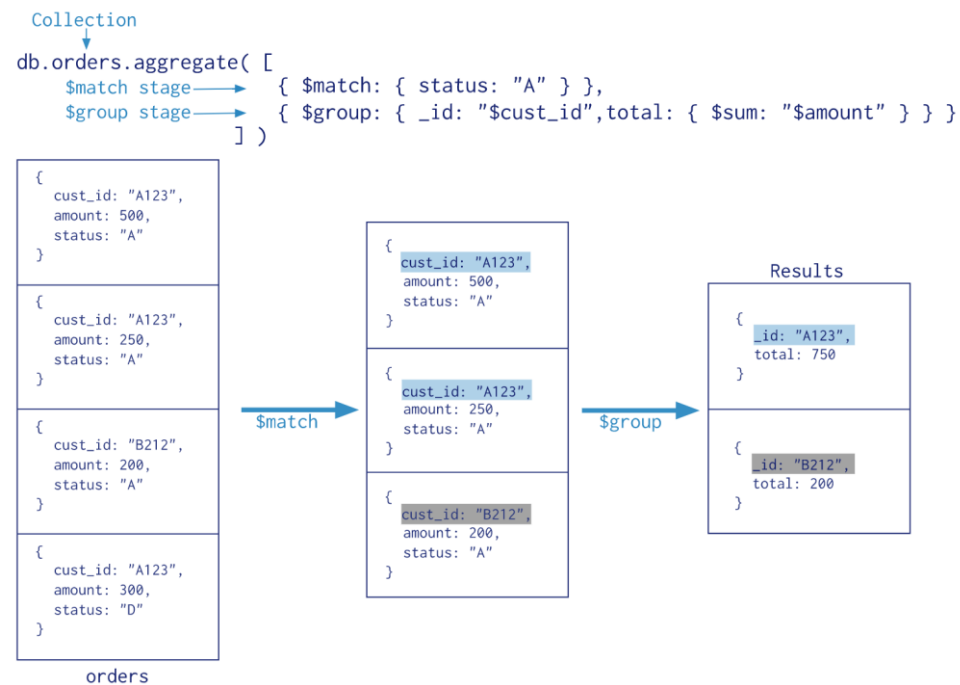
*[Mongo] db.collection.distinct("Field1", { Field2: Value2 })*

*[SQL] SELECT DISTINCT Field1 FROM collection WHERE Field2 = Value2*

# cRUD | Agrupación

```
db.collection.aggregate([ <stage1>, <stage2>, ... ])
```

<stage>: cada elemento o paso involucrado en la operación de agregación. El orden es muy importante, ya que el resultado de cada stage funciona como documento de entrada al siguiente stage. El único stage que tiene acceso al documento original (junto con todos los beneficios que esto implica) es el primero



[click to enlarge](#)

# CRUD | Agrupación

## Operaciones

- **\$match:** filtrado de resultados
- **\$group:** criterios de agrupación y resultados agregados
- **\$projection:** filtrado de campos y posibilidad de crear nuevos
- **\$sort:** ordenación de resultados
- **\$limit:** limitación de resultados
- **\$skip:** salto de resultados
- **\$bucket:** información agrupada por categorías y con estadísticas generadas
- **\$out:** resultado en nueva colección o en una ya existente

## Funciones de agregación

Función de agregación	Sintaxis
Suma/Cuenta	<code>{ \$sum: '\$field' } / { \$sum: 1 }</code>
Media	<code>{ \$avg: '\$field' }</code>
Mínimo/Máximo	<code>{ \$min: '\$field' } / { \$max: '\$field' }</code>

# cRUD | Más funciones | Ejemplo

1. ¿Cuántos documentos hay en la colección?
2. ¿Cuántos documentos tienen el campo **width** menor que 100?
3. Mostrar únicamente los campos **item** y **qty** de aquellos documentos que contenga el **tag** “orange”
4. Devolver todos los documentos ordenados por el campo **qty** descendente
5. ¿Cuántos valores distintos hay para el campo **item**?
6. Obtener la suma del campo **qty** de aquellos documentos que tengan los mismos tags

# cRUD | Más funciones | Ejemplo

*# Contar documentos de la colección*

```
db.TEST_COLLECTION.count()
```

*# Contar documentos con width < 100*

```
db.TEST_COLLECTION.count({ "size.width": { $lt: 100 } })
```

*# Mostrar campos item y qty de los que contenga tag orange*

```
db.TEST_COLLECTION.find({ tags: { $elemMatch: { $eq: "orange" } } }, { item: 1, qty: 1 })
```

*# Ordenación por qty descendente*

```
db.TEST_COLLECTION.find().sort({ qty: -1 })
```

*# Distintos valores campo item*

```
db.TEST_COLLECTION.distinct("item")
```

*# Agrupación por tags y suma de qty*

```
db.TEST_COLLECTION.aggregate([  
  { $group: { _id: "$tags", totalQuantity: { $sum: "$qty" } } }  
])
```

# CRUD | Actualización de documentos

```
db.collection.updateOne(<filter>, <update>)  
db.collection.updateMany(<filter>, <update>)  
db.collection.replaceOne(<filter>, <update>)
```

<filter>: documento con los filtros a aplicar para seleccionar los documentos que se quieren actualizar

<update>: documento que contiene el conjunto de actualizaciones a llevar a cabo (acciones de actualización) sobre los documentos filtrados

*[Mongo] db.collection.updateMany({ }, { Field2: Value3 })*

*[SQL] UPDATE collection SET Field2 = Value3*



# CRUD | Actualización de documentos

## Acciones de actualización

Acción	Sintaxis
Establecer valor	<code>{ \$set: {&lt;key&gt;:&lt;value&gt;, ...} }</code>
Eliminar campo	<code>{ \$unset: {&lt;key&gt;:""}, ...} }</code>
Renombrar campo	<code>{ \$rename: { "&lt;key&gt;": "&lt;newKey&gt;", ...} }</code>
Añadir valor a array (con duplicados)	<code>{ \$push: {&lt;key&gt;:&lt;value&gt;, ...} }</code>
Añadir valor a array (sin duplicados)	<code>{ \$addToSet: {&lt;key&gt;:&lt;value&gt;, ...} }</code>
Eliminar valor de array	<code>{ \$pull: {&lt;key&gt;:&lt;value&gt;, ...} }</code>
Eliminar último o primer elemento de array	<code>{ \$pop: {&lt;key&gt;:&lt;1/-1&gt;} }</code>
Incrementar/decrementar valor de campo	<code>{ \$inc: {&lt;key&gt;:&lt;num&gt;} }</code>

# CRUD | Actualización de documentos | Ejemplo

1. Renombrar el campo **item** por **name**
2. Añadir un nuevo **tag** a todos los documentos con el valor **black** SIN duplicar en aquellos que ya lo tengan
3. Incrementar en **10** unidades el valor del campo **qty**

# CRUD | Actualización de documentos | Ejemplo

*# Rename item by name*

```
db.TEST_COLLECTION.updateMany({}, { $rename: { "item": "name" } })
```

*# Add new black tag without duplicates*

```
db.TEST_COLLECTION.updateMany({}, { $addToSet: { tags: "black" } })
```

*# Inc by 10 the qty field*

```
db.TEST_COLLECTION.updateMany({}, { $inc: { qty: 10 } })
```

# CRUD | Eliminación de documentos

```
db.collection.deleteOne(<filter>)
```

```
db.collection.deleteMany(<filter>)
```

<filter>: documento con los filtros a aplicar para seleccionar los documentos que se quieren eliminar

*[Mongo] db.collection.deleteOne({ Field2: Value3 })*

*[SQL] DELETE FROM collection WHERE Field2 = Value3*

# CRUD | Eliminación de documentos | Ejemplo

1. Eliminar todos los documentos que **no** tengan nombre “AFI”
2. Borrar la colección

# CRUD | Eliminación de documentos | Ejemplo

*# All documents except AFI*

```
db.TEST_COLLECTION.deleteMany({ name: { $ne: "AFI" } })
```

*# Remove collection*

```
db.TEST_COLLECTION.drop()
```

# Índices

Se pueden crear índices para un único campo, para varios campos combinados e incluso índices espaciales para las geometrías

- **Mostrar índices**

```
db.collection.getIndexes()
```

- **Crear índice**

```
db.collection.createIndex({ <key1> : 1, <key2> : -1 })
```

El flag 1/-1 permite indicar si el índice se debe generar de forma ascendente o descendente

- **Borrar índice**

```
db.collection.dropIndex({ <key> : 1 })
```

- **Índice de texto**

```
db.collection.createIndex({ <key> : "text" })
```

Permite evitar el uso de expresiones regulares, y usarlo como tal mediante su combinación con el comando **\$search** en las funciones *find()* que se apliquen sobre el campo indexado como texto

- **Índice espacial**

```
db.collection.createIndex({ <key> : "2dsphere" })
```

Se puede indicar el tipo de índice, como: *unique*, *partial*, *sparse*...

# Geometrías espaciales

- MongoDB admite información y procesamiento geográfico
- Presenta mejor rendimiento que RDBMS con soporte geoespacial
- Gestiona los datos por medio de:
  - Índices
  - Limitado a polígonos válidos y sin cruces
  - Utiliza SRID EPSG:4326
  - Permite buscar: intersecciones, inclusiones, cercanía, dentro de...
- Admite objetos **GeoJSON**

```
<field>: {  
  type: <GeoJSON type> ,  
  coordinates: <coordinates>  
}
```

- Point / Multipoint
- LineString/ MultiLineString
- Polygon / MultiPolygon
- GeometryCollection



# Geometrías espaciales

Las geometrías tipo **círculo** y **rectángulo** no tienen representación pura, pero se representan mediante los siguientes símiles:

## Círculo

```
$geometry: {  
  type: "Point",  
  coordinates: [-3.489273, 40.583927]  
},  
$maxDistance: 100
```

## Polígono

Representación especial en base a las esquinas Suroeste (SW) y Noreste (NE)

```
$box: [[-3.895732, 40.589372], [-3.589327, 40.589327]]
```

# Consultas espaciales - \$near

Devuelve las geometrías que se encuentran cercanas a un punto dado

```
{  
  <location field>: {  
    $near: {  
      $geometry: {  
        type: "Point" ,  
        coordinates: [ <longitude> , <latitude> ]  
      },  
      $maxDistance: <distance in meters>,  
      $minDistance: <distance in meters>  
    }  
  }  
}
```

⚠ Los documentos devueltos vienen ordenados por distancia

**<location field>**: Es el campo con el índice 2dsphere

**\$geometry**: Es la definición del punto

**\$maxDistance**: Es el radio del círculo de búsqueda

**\$minDistance**: Es un campo opcional y nos permitiría búsquedas en una “rosquilla”

# Consultas espaciales - \$geoIntersects

Encuentra la intersección entre geometrías. La intersección se da cuando uno de los puntos de la geometría de los elementos buscados pertenece a la geometría pasada como parámetro

```
{
  <location field>: {
    $geoIntersects: {
      $geometry: {
        type: "xxxx" ,
        coordinates: [ xxxx ]
      }
    }
  }
}
```

**<location field>**: Es el campo con el índice 2dsphere

**\$geometry**: Es la definición de la geometría por la que se busca

# Consultas espaciales - \$geoWithin (I)

Caso particular de la intersección – geometrías contenidas dentro de otra geometría pasada por referencia

```
{  
  <location field>: {  
    $geoWithin: {  
      $geometry: {  
        type: "xxxx" ,  
        coordinates: [ xxxx ]  
      }  
    }  
  }  
}
```

**<location field>**: Es el campo con el índice 2dsphere

**\$geometry**: Es la definición de la geometría por la que se busca

# Consultas espaciales - \$geoWithin (II)

## Rectángulo

```
{
  <location field>: {
    $geoWithin: {
      $box: [SW, NE]
    }
  }
}
```

## Polígono

```
{
  <location field>: {
    $geoWithin: {
      $polygon: [[lon, lat], [lon, lat], ...]
    }
  }
}
```

## Círculo

```
{
  <location field>: {
    $geoWithin: {
      $center: [[lon, lat], radius]
    }
  }
}
```

# Ejercicios Prácticos

# Algunas preguntas

**Dataset:** [https://drive.google.com/file/d/1gP7M9o7\\_Lpv9CFGvIR1IGfYsWcvRD3NF/view?usp=sharing](https://drive.google.com/file/d/1gP7M9o7_Lpv9CFGvIR1IGfYsWcvRD3NF/view?usp=sharing)

1. ¿Cuántos índices hay en la colección?
  - Crear índice de tipo texto para el campo categories
  - Crear índice para el campo postal\_code
  - ¿Cuántos índices hay en la colección?
2. ¿Cuántos documentos hay en la colección?
3. ¿Qué negocios tienen como código postal el 97210?
4. ¿Cuántos negocios permiten la entrada de perros?
5. ¿Cuál es nombre del negocio con mejor puntuación y mayor número de opiniones?

# Algunas preguntas

6. ¿Cuántos códigos postales diferentes hay en la muestra?
7. ¿Cuántos negocios hay en cada ciudad?
8. ¿Cuántos negocios hay en torno al punto (-81.564481, 28.468456) en 5 kilómetros?  
(**PASOS**):
  - Crear una nueva colección generando el campo location de tipo GeoJSON a partir de las coodenadas
  - Generar el índice geoespacial en esta nueva colección y para el nuevo campo
  - Realizar la consulta pertinente

Referencia dataset: [https://www.kaggle.com/yelp-dataset/yelp-dataset?select=yelp\\_academic\\_dataset\\_business.json](https://www.kaggle.com/yelp-dataset/yelp-dataset?select=yelp_academic_dataset_business.json)



# Soluciones

1.

Query	Resultado
<code>db.TestMDS.getIndexes()</code>	1 índice ( <code>_id_</code> )
<code>db.TestMDS.createIndex({ categories : "text" })</code>	
<code>db.TestMDS.createIndex({ postal_code : 1 })</code>	
<code>db.TestMDS.getIndexes()</code>	3 índices ( <code>_id_</code> , <code>categories_text</code> , <code>postal_code_1</code> )

2.

Query	Resultado
<code>db.TestMDS.count()</code>	2425

3.

Query	Resultado
<code>db.TestMDS.find({ postal_code: "97210" })</code>	Se debe obtener la información completa de 7 documentos

# Soluciones

4.

Query	Resultado
<code>db.TestMDS.find({ "attributes.DogsAllowed": "True" }).count()</code>	104

5.

Query	Resultado
<code>db.TestMDS.aggregate([   { \$sort: { stars: -1, review_count: -1 } },   { \$limit: 1 },   { \$project: { name: 1, _id: 0 } } ])</code>	Mudra Massage

6.

Query	Resultado
<code>db.TestMDS.distinct("postal_code").length</code>	608

# Soluciones

7.

Query	Resultado
<pre>db.TestMDS.aggregate([   { \$group: {     _id: { cityName: "\$city" },     numBusiness: { \$sum: 1 }   }} ])</pre>	<p>Listado de documentos con el nombre de la ciudad y el número de negocios en cada una</p> <pre>/* 1 */ {   "_id" : {     "cityName" : "Mission Hill"   },   "numBusiness" : 1.0 }  /* 2 */ {   "_id" : {     "cityName" : "Gresham"   },   "numBusiness" : 3.0 }  /* 3 */ {   "_id" : {     "cityName" : "Saint Cloud"   },   "numBusiness" : 1.0 }</pre>

# Soluciones

8.

Query	Resultado
<pre>db.TestMDS.aggregate([   { \$project:     {       _id: 0, name: 1, stars: 1, review_count: 1,       hasWifi: "\$attributes.WiFi",       hasHappyHour: "\$attributes.HappyHour",       hasDelivery: "\$attributes.RestaurantesDelivery",       location: {         type: "Point",         coordinates: [ "\$longitude", "\$latitude" ]       }     }   },   { \$out: "newTestMDS" } ])</pre>	<p>Nueva colección con campos seleccionados y campo geométrico</p> <pre>{   "_id" : ObjectId("604e313dad5d0b483f95bff9"),   "name" : "Dekar Blues Taproom",   "stars" : 4.0,   "review_count" : 86,   "hasWifi" : "u'free'",   "hasHappyHour" : "True",   "location" : {     "type" : "Point",     "coordinates" : [       -105.2833461,       40.0175444     ]   } }</pre>
<pre>db.newTestMDS.createIndex({ location: "2dsphere" })</pre>	
<pre>db.newTestMDS.find({   location: {     \$near: {       \$geometry: {         type: "Point",         coordinates: [-81.564481, 28.468456]       },       \$maxDistance: 5000     }   } }).count()</pre>	3

# MongoDB desde Python

# ¿Por qué Python?

- Al ser un lenguaje de **tipado dinámico** encaja perfectamente con el paradigma **schema-less** de MongoDB
- Excelente manejo de diccionarios
- Compatibilidad de librerías Python con MongoDB
- Driver recomendado: **pymongo**
- En el IDE de Python que tengamos instalado – preferiblemente Spyder o PyCharm
  - Comprobar que tenemos pymongo instalado ejecutando ***import pymongo***
  - Si no está instalado, ejecutar ***conda install pymongo***

# Practicando con Python y MongoDB

- Establecer conexión con las bases de datos de MongoDB desde Python
- Trabajar con las bases de datos
- Trabajar con las colecciones y los documentos
  - Limpieza de datos
  - Estandarización de información
  - Uso de *pandas*
  - Creación de índices

<https://pymongo.readthedocs.io/en/stable/tutorial.html>

# Practicando con Python y MongoDB - Enunciados

## Dataset:

[https://drive.google.com/file/d/1XHMXS\\_Kc\\_7sXyquH4yrlbTQnxdhZfWlV/view?usp=sharing](https://drive.google.com/file/d/1XHMXS_Kc_7sXyquH4yrlbTQnxdhZfWlV/view?usp=sharing)

- Conexiones

<https://drive.google.com/file/d/1ZoK8dZaIB7VTgzF3xNJnndN1qJQ-AS1u/view?usp=sharing>

- Colecciones y documentos

<https://drive.google.com/file/d/1RK8cUyKiY3H5s1zk0we76RaBRQjtt5j4/view?usp=sharing>

- Trabajar con los datos

[https://drive.google.com/file/d/1Y5MqduGNL05dRk3kyjEZJsc68XIX-bj\\_/view?usp=sharing](https://drive.google.com/file/d/1Y5MqduGNL05dRk3kyjEZJsc68XIX-bj_/view?usp=sharing)



# Practicando con Python y MongoDB - Soluciones

- Conexiones

<https://drive.google.com/file/d/1dzU1PYthZ0TwC-qc5aF9KF6cKcW2UpMQ/view?usp=sharing>

- Colecciones y documentos

<https://drive.google.com/file/d/1gOdeWzoJtAug8glQAvAyx3bHHX5C2ipj/view?usp=sharing>

- Trabajar con los datos

[https://drive.google.com/file/d/1hFoZebem3hIFNf-iCZjBFU5nVA\\_tKEqP/view?usp=sharing](https://drive.google.com/file/d/1hFoZebem3hIFNf-iCZjBFU5nVA_tKEqP/view?usp=sharing)

# Práctica Python - Resumen

- Crear la conexión

```
from pymongo import MongoClient
conn = r'mongodb+srv://afidbuser2021:<password>@afi2021.i6cqp.mongodb.net/test'
connMongo = MongoClient(conn)
```

- Probar la conexión listando bases de datos

```
for dbname in connMongo.list_database_names():
    print(dbname)
```

- Acceder a una base de datos

```
# Example 1
connMongo.DATABASE_NAME
# Example 2
connMongo['DATABASE_NAME']
# Example 3
connMongo.get_database('DATABASE_NAME')
```

# Práctica Python - Resumen

- Listar colecciones

```
for colname in connMongo.DATABASE_NAME.list_collection_names():  
    print(colname)
```

- Crear y eliminar colecciones

```
# Create collection  
connMongo.DATABASE_NAME.create_collection('COLLECTION_NAME')  
# Drop collection  
connMongo.DATABASE_NAME.drop_collection('COLLECTION_NAME')
```

- Consultar dentro de colecciones

```
mydb = connMongo.DATABASE_NAME  
mycol = mydb.COLLECTION_NAME  
# Find first document that matches conditions or None  
mycol.find_one({ 'field1': 'value1' })  
# Find all documents that match conditions  
mycol.find({ 'field1': 'value1' })
```

# Práctica Python - Resumen

- Insertar documentos

```
# Single insert
doc = <objecto tipo diccionario>
mycol.insert_one(doc)
# Multiple insert
docs = [<objecto tipo diccionario>, <objecto tipo diccionario 2>, ... ]
mycol.insert_many(docs)
```

- Importar fichero json

```
with open('file.json', encoding = 'filenconding') as f:
    datastore = json.load(f)
mydb.COLLECTION_NAME.insert_many(datastore)
```

# Práctica Python - Resumen

- Actualizar documentos.

```
# Update documents to add new value
mycol.update_many({ 'field': 'oldValue' }, { '$set': { 'field': 'newValue' } })
# Update documents to remove value
mycol.update_many({ }, { '$unset': { 'field1': 1, 'field2': 1 } })
# Update one document based on id
for i in range(len(idList)):
    mycol.update_one({'_id': ObjectId(idList[i])},
                    {'$set':
                     {'field1': list1[i],
                      'field2': list1[i]
                     }
                    })
```

- Crear índices

```
mycol.create_index([('fieldToIndex', TYPE)])
```

# Referencias

# Referencias

- Documentación oficial de MongoDB  
<https://docs.mongodb.com/>
- Tutoriales de MongoDB  
<https://www.tutorialspoint.com/mongodb/>
- Tutorial pymongo  
<https://pymongo.readthedocs.io/en/stable/tutorial.html>
- Datasets de ejemplo  
<https://www.kaggle.com/>



Afi Escuela

---

© 2021 Afi Escuela. Todos los derechos reservados.