Facultad de Informática – Universidad Nacional de La Plata
Cátedra de Bases de Datos II

Bases de Datos 2 2022 -TP3

Bases de Datos NoSQL / Práctica con MongoDB

entrega: 13/6

Parte 1: Bases de Datos NoSQL y Relacionales

▶ Si bien las BBDD NoSQL tienen diferencias fundamentales con los sistemas de BBDD Relacionales o RDBMS, algunos conceptos comunes se pueden relacionar. Responda las siguientes preguntas, considerando MongoDB en particular como Base de Datos NoSQL.

1. ¿Cuáles de los siguientes conceptos de RDBMS existen en MongoDB? En caso de no existir, ¿hay alguna alternativa? ¿Cuál es?
    - Base de Datos
    - Tabla / Relación
    - Fila / Tupla
    - Columna

● Base de Datos
○ El concepto de Bases de Datos en MongoDB es el mismo que el concepto de Base de Datos en las RDBMS
● Tabla / Relación
○ Las colecciones de las bases de datos como MongoDb, equivalen a las tablas en las bases de datos relacionales, pero esto no quiere decir que se usen de la misma forma, o se relacionen entre sí como las “bases de datos relaciones”.
● Fila / Tupla
○ Mongo puede guardar array de elementos, que se denominan documentos. Un documento correspondería a una tupla. El documento es la unidad mínima de las bases de datos como MongoDB. Los elementos que guarda un array pueden ser de cualquier tipo, es decir que pueden ser strings, números, otros arrays o incluso subdocumentos. Cada documento puede tener su propia estructura.
● Columna
○ En mongo no existe el concepto de columna, pero equivaldría al concepto de campo, un documento contiene campos. La gran diferencia es que como cada documento puede tener su propia estructura, los documentos que representan al mismo tipo de objetos, es decir, que se almacenan en la misma colección, no necesitan contener los mismos campos.

RDBMS	MongoDB
Base de datos	Base de datos
Tabla	Colección
Fila	Documento
Columna	Campo

2. MongoDB tiene soporte para transacciones, pero no es igual que el de los RDBMS. ¿Cuál es el alcance de una transacción en MongoDB?

Cuando una transacción hace commit, todos los cambios hechos a los datos en la transacción se guardan y son visibles fuera de la transacción. Antes de la versión 4.0 MongoDB garantizaba transacciones ACID a nivel de documento, es decir qué operaciones realizadas entre subdocumentos dentro de un documento cumplian con la condicion de que si fallaba, la propia base de datos se encargaba de hacer un rollback al estado anterior al inicio de la operación. Sin embargo existen casos de uso en los cuales es absolutamente necesario contar con transaccionalidad entre documentos. Debido a esto en su versión 4.0 MongoDB implementa las transacciones ACID entre documentos

3. Para acelerar las consultas, MongoDB tiene soporte para índices. ¿Qué tipos de índices soporta?

MongoDB soporta los siguientes tipos de índices:

- Single Field Index: Este tipo de índice se aplica sobre un campo de un documento. Permite ordenar una colección en base a dicho campo y su sort-criteria (Ascendente o Descendente), con lo que las búsquedas por igualdad o rangos se vuelve muy eficiente.

- Compound Index: Este tipo de índices permite extender la idea del “Single Field Index” a más de un campo, es decir que ahora podemos hacer índices compuestos por más de una clave. Por cada campo del índice compuesto se indica el sort-criteria, y para efectuar la construcción del mismo MongoDB respeta el orden de campos indicado por el programador. Es entonces que el orden de los campos a indexar se vuelve importante, ya que con otro orden la estructura resultante podría no ser de utilidad para el caso de uso en desarrollo.

- Multikey Index: En ciertos escenarios, puede ocurrir que un campo tenga como valor asociado un Array de valores, y sea necesario buscar todos los documentos que contengan ciertos elementos dentro de dicho campo de tipo Array. MongoDB introduce los índices multi-clave, que permiten indexar la información de campos de tipo Array, para su posterior búsqueda eficiente.

- Geospatial Index: Para admitir consultas eficientes de datos de coordenadas geoespaciales, MongoDB proporciona dos índices especiales: índices 2d que usan geometría plana al devolver resultados e índices 2dsphere que usan geometría esférica para devolver resultados.

- Text Indexes: Este tipo de índices permite la búsqueda de strings dentro de una colección,

dotando al motor de la posibilidad de realizar búsquedas de texto eficientes, ignorando en su proceso de indexación a palabras vacías, o realizando “Stemming”, un proceso por el cual se toman las palabras de un texto y se las lleva a su forma “Madre” o raiz, evitando conjugaciones y demás agregados sobre dicha palabra. Esto permite que varias palabras expresadas de formas distintas pero que comparten el mismo “Stem” puedan ser consideradas sinónimos.

- Hashed Indexes: Este tipo de índices se utiliza para dar soporte al sharding basado en Hashing. Lo que hacen estos índices es indexar el hash del valor del campo usado como clave. Como beneficio, permite hacer una especie de “load-balancing” de cada shard, evitando que claves muy próximas o similares siempre residan en el mismo shard, y agregando un nivel de dispersión mayor a las claves. Como contrapartida, hashing no permite búsquedas por rango, solo de acceso directo (por igualdad).

4. ¿Existen claves foráneas en MongoDB?

Mongo no tiene claves foráneas, pero existen 2 formas para poder relacionar documentos:

● Podemos guardar en un campo de un documento, el “_id” de otro documento para

poder referenciar y obtenerlo a través de una segunda query.

● DBRefs: son convenciones para representar un documento. Incluyen el nombre de la colección y el id del campo (también pueden tener el nombre de la base de datos si es necesario). Se escriben así:

```
{ "$ref" : <value>, "$id" : <value>, "$db" : <value> }
```

DBRefs son referencias de un documento a otro utilizando el valor del campo del primer documento _id, nombre de la colección (y opcionalmente el nombre de base de datos). Con esto los DBRefs permiten vincular documentos de varias colecciones para linkarse en una sola colección. Los DBRefs proporcionan en esencia una semántica común para la representación de los vínculos entre documentos. Los DBRefs también requieren consultas adicionales para devolver los documentos de referencia. Por otro lado la mayoría de los drivers ofrecen métodos de utilidad para hacer la query de la DBRef de forma automática.

2. MongoDB tiene soporte para transacciones, pero no es igual que el de los RDBMS. ¿Cuál es el alcance de una transacción en MongoDB?

No entendí la pregunta

Las transacciones de MongoDB, al usar el modelo NOSQL, tiene las características BASE:

Basicaly **A**vailable: Dado el caso de que haya fallos que eviten el acceso a algún segmento de datos, esto no resulta necesariamente en que se caiga toda la base de datos.

Soft state: Se abandona el concepto de Consistencia de ACID. La consistencia de los datos es un problema del desarrollador y no debería ser responsabilidad de la DBMS.

Eventually consistent: Los datos eventualmente van a converger en un estado consistente (en contraposición con ACID que requiere una consistencia inmediata). Esto implica que se pueden comitear transacciones en orden no secuencial.

3. Para acelerar las consultas, MongoDB tiene soporte para índices. ¿Qué tipos de índices soporta?

- Single file: son creados por el usuario (a excepción de “_id”) y se puede ascender o descender.
- Compound index: son creados por el usuario y las búsquedas se realizan primero por el primer índice, después dentro del primer índice con el segundo, y así con el n-esimo dentro del n-esimo+1.
- Multikey: son creadas por mongo automáticamente para indexar el contenido de un array, automáticamente. Solo se usan si estamos indexando un archivo que tiene un array.
- Geospatial: se utiliza para indexar datos ya sea en un plano 2d o en el plano 2d sobre una esfera.
- Text index: son como single file pero utilizando strings en vez de cualquier valor por el cual se pueda ascender o descender.
- Hashed index: Mongo usa estos índices para separar sus shards.

4. ¿Existen claves foráneas en MongoDB?

No precisamente, lo más similar que existe son las referencias o DBRefs. Las DBRef son convenciones para representar un documento. Incluyen el nombre de la colección y el id del campo (también pueden tener el nombre de la base de datos si es necesario). Se escriben así:

```
{ "$ref" : <value>, "$id" : <value>, "$db" : <value> }
```

Parte 2: Primeros pasos con MongoDB

▶ Descargue la última versión de MongoDB desde el sitio oficial. Ingrese al cliente de línea de comando para realizar los siguientes ejercicios.

5. Cree una nueva base de datos llamada **vaccination**, y una colección llamada **nurses**. En esa colección inserte un nuevo documento (una enfermera) con los siguientes atributos:

```
{name:'Morella Crespo', experience:9}
```

recupere la información de la enfermera usando el comando `db.nurses.find()` (puede agregar la función `.pretty()` al final de la expresión para ver los datos indentados). Notará que no se encuentran exactamente los atributos que insertó. ¿Cuál es la diferencia? .

5. use vaccination
db.nurses.insert({name:"Morella Crespo", experience:9})
db.nurses.find() devuelve, además de los datos insertados, el campo _id que se genera automáticamente.

6. db.nurses.insert([{name:"Gale Molina", experience:8, vaccines:["AZ", "Moderna"]},
{name:"Honoria Fernández", experience:5, vaccines:["Pfizer", "Moderna", "Sputnik V"]},
{name:"Gonzalo Gallardo", experience:3}, {name:"Altea Parra", experience:8, vaccines:["Pfizer"]}])

6.a) db.nurses.find({experience: {$lte: 5}})

6.b) db.nurses.find({vaccines: “Pfizer”})
6.c) db.nurses.find({vaccines: {$exists: false}})
6.d) db.nurses.find({name: {$regex: /Fernández$/ }})
6.e) db.nurses.find({experience: {$gte: 6}, vaccines: “Moderna”})
6.f) db.nurses.find({experience: {$gte: 6}, vaccines: “Moderna”}, {_id:0, name:1})

7. db.nurses.updateOne({name:”Gonzalo Gallardo”}, {$set: {vaccines: []}})

8. db.nurses.updateOne({name:”Gale Molina”}, {$set: {experience: 9}})

9. db.nurses.updateOne({name:”Altea Parra”}, {$push: {vaccines: “AZ”}})

10. db.nurses.updateMany({vaccines: “Pfizer”}, {$mul: {experience: 2}})

Facu Cadaa

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use vaccination
switched to db vaccination
> db.nurses.insert({name:"Morella Crespo", experience:9})
WriteResult({ "nInserted" : 1 })
> db.nurses.find().pretty()
{
        "_id" : ObjectId("629e5df41ab45e53ac45c2d9"),
        "name" : "Morella Crespo",
        "experience" : 9
}
> _
```

La diferencia es que contiene el campo _id, que se genera automáticamente

Joaquin Galdeano

```
> db.nurses.find()
< { _id: ObjectId("629a0bf5adfdd3e1544fc34e"),
    name: 'Morella Crespo',
    experience: 9 }
```

Además de los atributos insertados, también devuelve “_id” generado automáticamente por MongoDB.

Fran Camacho

creamos la base de datos

```
> use vaccination
switched to db vaccination
```

creamos la colleccion

```
> db.createCollection("nurses")
{ "ok" : 1 }
```

insertamos un elemento

```
> db.nurses.insert({name:'Morella Crespo', experience:9})
WriteResult({ "nInserted" : 1 })
```

se recupera la información

```
> db.nurses.find().pretty()
{
        "_id" : ObjectId("629d3d0b5bcf0e0b4e77e33a"),
        "name" : "Morella Crespo",
        "experience" : 9
}
```

Además de los atributos insertados, MongoDB agrega un identificador al documento ingresado. El identificador se reconoce con el nombre de “_id”, junto con un hash asociado.

▶ Una característica fundamental de MongoDB y otras bases NoSQL es que los documentos no tienen una estructura definida, como puede ser una tabla en un RDBMS. En una misma colección pueden convivir documentos con diferentes atributos, e incluso atributos de múltiples valores y documentos embebidos.

6. Agregue los siguientes documentos a la colección de enfermeros:

```
{name:’Gale Molina', experience:8, vaccines: ['AZ', ‘Moderna']}
{name:’Honoria Fernández', experience:5, vaccines: ['Pfizer', ‘Moderna’, ‘Sputnik V’]} {name:’Gonzalo Gallardo', experience:3}
{name:’Altea Parra', experience:6, vaccines: ['Pfizer']}
```

Facu Cadaa

```
> db.nurses.insert([{name:"Gale Molina", experience:8, vaccines:["AZ", "Moderna"]}, {name:"Honoria Fernández", experience:5, vaccines:["Pfizer"
, "Moderna", "Sputnik V"]}, {name:"Gonzalo Gallardo", experience:3}, {name:"Altea Parra", experience:8, vaccines:["Pfizer"]} ] )
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 4,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

Franco Camacho

```
> db.nurses.insertOne({name:'Gale Molina', experience:8, vaccines: ['AZ', 'Moderna']})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("629d41885bcf0e0b4e77e33b")
}
```

```
> db.nurses.insertOne({name:'Honoria Fernández', experience:5, vaccines: ['Pfizer', 'Moderna', 'Sputnik V']})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("629d44550330fa60ab48b954")
}
```

```
> db.nurses.insertOne({name:'Gonzalo Gallardo', experience:3})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("629d44810330fa60ab48b955")
}
```

```
> db.nurses.insertOne({name:'Altea Parra', experience:6, vaccines: ['Pfizer']})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("629d45950330fa60ab48b956")
}
```

Y busque los enfermeros:

- de 5 años de experiencia o menos

Facundo Cadaa

```
> db.nurses.find( { experience: { $lte: 5 } } ).pretty()
{
        "_id" : ObjectId("629e8f8c1ab45e53ac45c2db"),
        "name" : "Honoria Fernández",
        "experience" : 5,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
{
        "_id" : ObjectId("629e8f8c1ab45e53ac45c2dc"),
        "name" : "Gonzalo Gallardo",
        "experience" : 3
}
```

- Joaquin Galdeano

```
> db.nurses.find({experience: {$lte: 5}})
{ _id: ObjectId("629a1047adfdd3e1544fc358"),
  name: 'Gonzalo Gallardo',
  experience: 3,
  vaccines: [] }
```

- Fran Camacho

```
> db.nurses.find({experience:{$lte:5}}).pretty()
{
        "_id" : ObjectId("629d44550330fa60ab48b954"),
        "name" : "Honoria Fernández",
        "experience" : 5,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
{
        "_id" : ObjectId("629d44810330fa60ab48b955"),
        "name" : "Gonzalo Gallardo",
        "experience" : 3
}
```

- que hayan aplicado la vacuna “Pfizer”

Facundo Cadaa

```
}
> db.nurses.find( { vaccines: { $all: ["Pfizer"] } } ).pretty()
{
        "_id" : ObjectId("629e8f8c1ab45e53ac45c2db"),
        "name" : "Honoria Fernández",
        "experience" : 5,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
{
        "_id" : ObjectId("629e8f8c1ab45e53ac45c2dd"),
        "name" : "Altea Parra",
        "experience" : 8,
        "vaccines" : [
                "Pfizer"
        ]
}
>
```

- Joaquin Galdeano

```
> db.nurses.find({ vaccines: "Pfizer"})
< { _id: ObjectId("629a1041adfdd3e1544fc356"),
    name: 'Honoria Fernández',
    experience: 5,
    vaccines: [ 'Pfizer', 'Moderna', 'Sputnik V' ] }
  { _id: ObjectId("629a104fadfdd3e1544fc35a"),
    name: 'Altea Parra',
    experience: 6,
    vaccines: [ 'Pfizer' ] }
```

-Fran Camacho

```
> db.nurses.find({vaccines:'Pfizer'}).pretty()
{
        "_id" : ObjectId("629d44550330fa60ab48b954"),
        "name" : "Honoria Fernández",
        "experience" : 5,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
{
        "_id" : ObjectId("629d45950330fa60ab48b956"),
        "name" : "Altea Parra",
        "experience" : 6,
        "vaccines" : [
                "Pfizer"
        ]
}
```

- que no hayan aplicado vacunas (es decir, que el atributo *vaccines* esté ausente)
Facundo Cadaa

```
}
> db.nurses.find( { vaccines: { $exists : false } } ).pretty()
{
        "_id" : ObjectId("629e5df41ab45e53ac45c2d9"),
        "name" : "Morella Crespo",
        "experience" : 9
}
{
        "_id" : ObjectId("629e8f8c1ab45e53ac45c2dc"),
        "name" : "Gonzalo Gallardo",
        "experience" : 3
}
>
```

-Fran Camacho

```
> db.nurses.find({vaccines:{$exists: false}}).pretty()
{
        "_id" : ObjectId("629d43320330fa60ab48b952"),
        "name" : "Morella Crespo",
        "experience" : 9
}
{
        "_id" : ObjectId("629d44810330fa60ab48b955"),
        "name" : "Gonzalo Gallardo",
        "experience" : 3
}
```

Joaquin Galdeano

```
> db.nurses.find({vaccines: {$exists: false}})
< { _id: ObjectId("629a0bf5adfdd3e1544fc34e"),
    name: 'Morella Crespo',
    experience: 9 }
  { _id: ObjectId("629a1047adfdd3e1544fc358"),
    name: 'Gonzalo Gallardo',
    experience: 3 }
```

- de apellido ‘Fernández’

Facundo Cadaa

```
> db.nurses.find( { name: { $regex : /Fernández$/ } } ).pretty()
{
        "_id" : ObjectId("629e8f8c1ab45e53ac45c2db"),
        "name" : "Honoria Fernández",
        "experience" : 5,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
>
```

Franco Camacho

```
> db.nurses.find({"name": /.*Fernández.*/}).pretty()
{
        "_id" : ObjectId("629d44550330fa60ab48b954"),
        "name" : "Honoria Fernández",
        "experience" : 5,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
```

- Joaquin Galdeano

```
> db.nurses.find({name: {$regex: "Fernández"}})
< { _id: ObjectId("629a1041adfdd3e1544fc356"),
    name: 'Honoria Fernández',
    experience: 5,
    vaccines: [ 'Pfizer', 'Moderna', 'Sputnik V' ] }
```

- con 6 o más años de experiencia y que hayan aplicado la vacuna ‘Moderna’

Facundo Cadaa

```
> db.nurses.find( { vaccines: { $all: ["Moderna"]},  experience: { $gte: 6 } } ).pretty()
{
        "_id" : ObjectId("629e8f8c1ab45e53ac45c2da"),
        "name" : "Gale Molina",
        "experience" : 8,
        "vaccines" : [
                "AZ",
                "Moderna"
        ]
}
>
```

Franco camacho

```
> db.nurses.find({vaccines:'Moderna', experience:{$gte: 6}}).pretty()
{
        "_id" : ObjectId("629d44150330fa60ab48b953"),
        "name" : "Gale Molina",
        "experience" : 8,
        "vaccines" : [
                "AZ",
                "Moderna"
        ]
}
```

Joaquin Galdeano

```
> db.nurses.find({experience: {$gte: 6}, vaccines: "Moderna"})
{ _id: ObjectId("629a0e8dadfdd3e1544fc354"),
  name: 'Gale Molina',
  experience: 8,
  vaccines: [ 'AZ', 'Moderna' ] }
```

vuelva a realizar la última consulta pero proyecte sólo el nombre del enfermero/a en los resultados, omitiendo incluso el atributo **_id** de la proyección.

Facundo Cadaa

```
> db.nurses.find( { vaccines: { $all: ["Moderna"]},  experience: { $gte: 6 } }, {_id:0, name: 1} ).pretty()
{ "name" : "Gale Molina" }
>
```

Franco Camacho

```
> db.nurses.find({vaccines:'Moderna', experience:{$gte: 6}}, {_id:0, name:1}).prett
y()
{ "name" : "Gale Molina" }
```

joaquin Galdeano

```
> db.nurses.find({experience: {$gte: 6}, vaccines: "Moderna"}, { name: 1, _id: 0})
< { name: 'Gale Molina' }
```

▶ En MongoDB hay diferentes maneras de realizar actualizaciones, de acuerdo a las necesidades del esquema flexible de documentos.

7. Actualice a “Gale Molina” cambiándole la experiencia a 9 años.

```
> db.nurses.updateOne( { name:"Gale Molina"}, {$set: {experience: 9}} )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.nurses.find( { name:"Gale Molina"})
{ "_id" : ObjectId("629e8f8c1ab45e53ac45c2da"), "name" : "Gale Molina",
"experience" : 9, "vaccines" : [ "AZ", "Moderna" ] }
>
```

Franco Camacho

```
> db.nurses.updateOne({name: "Gale Molina"}, {$set: {experience: 9}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.nurses.find().pretty()
{
        "_id" : ObjectId("629d43320330fa60ab48b952"),
        "name" : "Morella Crespo",
        "experience" : 9
}
{
        "_id" : ObjectId("629d44150330fa60ab48b953"),
        "name" : "Gale Molina",
        "experience" : 9,
        "vaccines" : [
                "AZ",
                "Moderna"
        ]
}
{
        "_id" : ObjectId("629d44550330fa60ab48b954"),
        "name" : "Honoria Fernández",
        "experience" : 5,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
{
        "_id" : ObjectId("629d44810330fa60ab48b955"),
        "name" : "Gonzalo Gallardo",
        "experience" : 3
}
{
        "_id" : ObjectId("629d45950330fa60ab48b956"),
        "name" : "Altea Parra",
        "experience" : 6,
        "vaccines" : [
                "Pfizer"
        ]
}
```

Joaquin Galdeano

```
> db.nurses.updateOne({name :"Gale Molina"}, {$set: {experience: 9}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

8. Cree el *array* de vacunas (*vaccines*) para ”Gonzalo Gallardo".

Facundo Cadaa

```
pdateOne( { name:"Gonzalo Gallardo"}, {$set: {vaccines: [
]}} )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCou
nt" : 1 }
> db.nurses.find( { name:"Gonzalo Gallardo"} )
{ "_id" : ObjectId("629e8f8c1ab45e53ac45c2dc"), "name" :
"Gonzalo Gallardo", "experience" : 3, "vaccines" : [ ] }
> _
```

```
> db.nurses.updateOne({name: "Gonzalo Gallardo"}, {$set: {vaccines: []}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.nurses.find().pretty()
{
        "_id" : ObjectId("629d43320330fa60ab48b952"),
        "name" : "Morella Crespo",
        "experience" : 9
}
{
        "_id" : ObjectId("629d44150330fa60ab48b953"),
        "name" : "Gale Molina",
        "experience" : 9,
        "vaccines" : [
                "AZ",
                "Moderna"
        ]
}
{
        "_id" : ObjectId("629d44550330fa60ab48b954"),
        "name" : "Honoria Fernández",
        "experience" : 5,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
{
        "_id" : ObjectId("629d44810330fa60ab48b955"),
        "name" : "Gonzalo Gallardo",
        "experience" : 3,
        "vaccines" : [ ]
}
{
        "_id" : ObjectId("629d45950330fa60ab48b956"),
        "name" : "Altea Parra",
        "experience" : 6,
        "vaccines" : [
                "Pfizer"
        ]
}
```

Joaquin Galdeano

```
> db.nurses.updateOne({name: "Gonzalo Gallardo"}, {$set: {vaccines: []}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

9. Agregue “AZ” a las vacunas de “Altea Parra”.

Facundo Cadaa

```
> db.nurses.updateOne( { name:"Altea Parra"}, {$push: {vaccines: "AZ"}}
)
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.nurses.find( { name:"Altea Parra"} )
{ "_id" : ObjectId("629e8f8c1ab45e53ac45c2dd"), "name" : "Altea Parra",
"experience" : 8, "vaccines" : [ "Pfizer", "AZ" ] }
>
```

Franco Camacho

```
> db.nurses.updateOne({name: "Altea Parra"}, {$push: {vaccines: "AZ"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.nurses.find().pretty()
{
        "_id" : ObjectId("629d43320330fa60ab48b952"),
        "name" : "Morella Crespo",
        "experience" : 9
}
{
        "_id" : ObjectId("629d44150330fa60ab48b953"),
        "name" : "Gale Molina",
        "experience" : 9,
        "vaccines" : [
                "AZ",
                "Moderna"
        ]
}
{
        "_id" : ObjectId("629d44550330fa60ab48b954"),
        "name" : "Honoria Fernández",
        "experience" : 5,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
{
        "_id" : ObjectId("629d44810330fa60ab48b955"),
        "name" : "Gonzalo Gallardo",
        "experience" : 3,
        "vaccines" : [ ]
}
{
        "_id" : ObjectId("629d45950330fa60ab48b956"),
        "name" : "Altea Parra",
        "experience" : 6,
        "vaccines" : [
                "Pfizer",
                "AZ"
        ]
}
```

Joaquin Galdeano

```
> db.nurses.updateOne({name: "Altea Parra"}, {$push: {"vaccines": "AZ"}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

10. Duplique la experiencia de **todos** los enfermeros que hayan aplicado la vacuna “Pfizer”

Facundo Cadaa

Franco Camacho

```
> db.nurses.updateMany({"vaccines": "Pfizer"}, {$mul: {experience:2}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
> db.nurses.find().pretty()
{
        "_id" : ObjectId("629d43320330fa60ab48b952"),
        "name" : "Morella Crespo",
        "experience" : 9
}
{
        "_id" : ObjectId("629d44150330fa60ab48b953"),
        "name" : "Gale Molina",
        "experience" : 9,
        "vaccines" : [
                "AZ",
                "Moderna"
        ]
}
{
        "_id" : ObjectId("629d44550330fa60ab48b954"),
        "name" : "Honoria Fernández",
        "experience" : 10,
        "vaccines" : [
                "Pfizer",
                "Moderna",
                "Sputnik V"
        ]
}
{
        "_id" : ObjectId("629d44810330fa60ab48b955"),
        "name" : "Gonzalo Gallardo",
        "experience" : 3,
        "vaccines" : [ ]
}
{
        "_id" : ObjectId("629d45950330fa60ab48b956"),
        "name" : "Altea Parra",
        "experience" : 12,
        "vaccines" : [
                "Pfizer",
                "AZ"
        ]
}
>
```

Joaquin Galdeano

```
> db.nurses.updateMany({vaccines: "Pfizer"}, {$mul: {experience: 2}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0 }
```

Parte 3: Índices

- Elimine a todos los enfermeros de la colección. Guarde en un archivo llamado ‘generador.js’ el siguiente código JavaScript y ejecútelo con: `load(<ruta del archivo ‘generador.js’>)`. Si utiliza un cliente que lo permita (ej. Robo3T), se puede ejecutar directamente en el espacio de consultas.

```
var vaccines = ['AZ', 'Pfizer', 'Moderna', 'Sputnik V',
"Johnson"]; for (var i = 1; i <= 2000; i++) {
	var randomVax = vaccines.sort( function() { return 0.5 -
Math.random() } ).slice(1, Math.floor(Math.random() * 5));
	var randomExperience = Math.ceil(2+(Math.random() * 20 - 2));
	db.nurses.insert({
		name:'Enfermero '+i,
		experience:randomExperience,
		tags: randomVax
	});}
var patientNumber = 0;
for (var i = 1; i <= 2000; i++) {
		var dosesCount = 50 + Math.ceil(Math.random() * 100);
		for (var r = 1; r <= dosesCount; r++){
			var randomLong = -34.56 - (Math.random() * .23);
			var randomLat = -58.4 - (Math.random() * .22);
			db.patients.insert({
			 name:'Paciente '+patientNumber,
			  address: {type: "Point",coordinates: [randomLat, randomLong]}
			});
			patientNumber++;
			var year = 2020 + Math.round(Math.random());
			var month = Math.ceil(Math.random() * 12);
			var day = Math.ceil(Math.random() * 28);
			db.doses.insert({
			 nurse:'Enfermero '+i,
			 patient:'Paciente '+patientNumber,
			  vaccine: vaccines[Math.floor(Math.random()*vaccines.length)],
			 date: new Date(year+'-'+month+'-'+day)
			});
		}}
```

11. Busque en la colección de compras (`doses`) si existe algún índice definido.

- Franco Camacho

```
> load('C:\\Users\\franco\\Desktop\\bd2\\generador.js')
true
```

```
> db.doses.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
```

- Joaquin Galdeano

```
> db.doses.getIndexes()
< [ { v: 2, key: { _id: 1 }, name: '_id_' } ]
```

El índice definido para ‘doses’ es el índice “_id” por defecto.

- Facundo Cadaa

```
> use vaccination
switched to db vaccination
> show collections
nurses
> db.nurses.remove({})
WriteResult({ "nRemoved" : 5 })
```

```
> load("C:\\Users\\User\\Facu_informatica\\Facu informatica\\2022\\BD2\\generador.js")
true
> load(\Users\User\Facu_informatica\Facu informatica\2022\BD2\generador.js)
SyntaxError: invalid escape sequence :
@(shell):1:5
> show collections
doses
nurses
patients
```

```
> db.doses.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
>
```

12. Cree un índice para el campo `nurse` de la colección `doses`. Busque las dosis que tengan en el nombre del enfermero el string “11” y utilice el método `explain("executionStats")` al final de la consulta, para comparar la **cantidad de documentos examinados** y el **tiempo en milisegundos** de la consulta con y sin índice.

- Franco Camacho

sin índice

```
> db.doses.find( {"nurse":/11/} ).explain("executionStats")
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "test.doses",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "nurse" : {
                                "$regex" : "11"
                        }
                },
                "maxIndexedOrSolutionsReached" : false,
                "maxIndexedAndSolutionsReached" : false,
                "maxScansToExplodeReached" : false,
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "nurse" : {
                                        "$regex" : "11"
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 17442,
                "executionTimeMillis" : 1119,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 332788,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "nurse" : {
```

```
                                        "$regex" : "11"
                                }
                        },
                        "nReturned" : 17442,
                        "executionTimeMillisEstimate" : 246,
                        "works" : 332790,
                        "advanced" : 17442,
                        "needTime" : 315347,
                        "needYield" : 0,
                        "saveState" : 334,
                        "restoreState" : 334,
                        "isEOF" : 1,
                        "direction" : "forward",
                        "docsExamined" : 332788
                }
        },
        "command" : {
                "find" : "doses",
                "filter" : {
                        "nurse" : /11/
                },
                "$db" : "test"
        },
        "serverInfo" : {
                "host" : "LAPTOP-L8EH9CHU",
                "port" : 27017,
                "version" : "5.0.9",
                "gitVersion" : "6f7dae919422dcd7f4892c10ff20cdc721ad00e6"
        },
        "serverParameters" : {
                "internalQueryFacetBufferSizeBytes" : 104857600,
                "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
                "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
                "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
                "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
                "internalQueryProhibitBlockingMergeOnMongoS" : 0,
                "internalQueryMaxAddToSetBytes" : 104857600,
                "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
        },
        "ok" : 1
}
```


con indice:

```
> db.doses.createIndex({nurse:1})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

```
> db.doses.find( {"nurse":/11/} ).explain("executionStats")
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "test.doses",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "nurse" : {
                                "$regex" : "11"
                        }
                },
                "maxIndexedOrSolutionsReached" : false,
                "maxIndexedAndSolutionsReached" : false,
                "maxScansToExplodeReached" : false,
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "filter" : {
                                        "nurse" : {
                                                "$regex" : "11"
                                        }
                                },
                                "keyPattern" : {
                                        "nurse" : 1
                                },
                                "indexName" : "nurse_1",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                        "nurse" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "nurse" : [
                                                "[\"\", {})",
                                                "[/11/, /11/]"
                                        ]
                                }
                        }
```

```
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 17442,
                "executionTimeMillis" : 1193,
                "totalKeysExamined" : 332788,
                "totalDocsExamined" : 17442,
                "executionStages" : {
                        "stage" : "FETCH",
                        "nReturned" : 17442,
                        "executionTimeMillisEstimate" : 244,
                        "works" : 332789,
                        "advanced" : 17442,
                        "needTime" : 315346,
                        "needYield" : 0,
                        "saveState" : 332,
                        "restoreState" : 332,
                        "isEOF" : 1,
                        "docsExamined" : 17442,
                        "alreadyHasObj" : 0,
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "filter" : {
                                        "nurse" : {
                                                "$regex" : "11"
                                        }
                                },
                                "nReturned" : 17442,
                                "executionTimeMillisEstimate" : 223,
                                "works" : 332789,
                                "advanced" : 17442,
                                "needTime" : 315346,
                                "needYield" : 0,
                                "saveState" : 332,
                                "restoreState" : 332,
                                "isEOF" : 1,
                                "keyPattern" : {
                                        "nurse" : 1
                                },
                                "indexName" : "nurse_1",
                                "isMultiKey" : false,
```

```
                                "multiKeyPaths" : {
                                        "nurse" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "nurse" : [
                                                "[\"\", {})",
                                                "[/11/, /11/]"
                                        ]
                                },
                                "keysExamined" : 332788,
                                "seeks" : 1,
                                "dupsTested" : 0,
                                "dupsDropped" : 0
                        }
                }
        },
        "command" : {
                "find" : "doses",
                "filter" : {
                        "nurse" : /11/
                },
                "$db" : "test"
        },
        "serverInfo" : {
                "host" : "LAPTOP-L8EH9CHU",
                "port" : 27017,
                "version" : "5.0.9",
                "gitVersion" : "6f7dae919422dcd7f4892c10ff20cdc721ad00e6"
        },
        "serverParameters" : {
                "internalQueryFacetBufferSizeBytes" : 104857600,
                "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
                "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
                "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
                "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
                "internalQueryProhibitBlockingMergeOnMongoS" : 0,
                "internalQueryMaxAddToSetBytes" : 104857600,
                "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
        },
        "ok" : 1
}
```


- Joaquin Galdeano

```
executionStats:
 { executionSuccess: true,
   nReturned: 12286,
   executionTimeMillis: 725,
   totalKeysExamined: 0,
   totalDocsExamined: 199168,
```

Sin utilizar índices, en la consulta (a) por dosis con enfermeros que contengan el string “11” (db.doses.find({nurse: {$regex: “11”}})) se examinaron 199168 documentos en 725 milisegundos.

Creación de índice por campo nurse en la colección doses:

```
> db.doses.createIndex( {nurse: -1})
< 'nurse_-1'
```

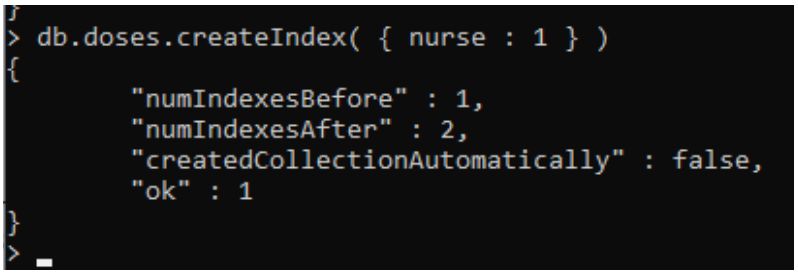
Con el nuevo índice creado, haciendo la misma consulta (a), se obtiene que se examinaron 12286 documentos en 202 milisegundos.

```
executionStats:
 { executionSuccess: true,
   nReturned: 12286,
   executionTimeMillis: 202,
   totalKeysExamined: 199168,
   totalDocsExamined: 12286,
```

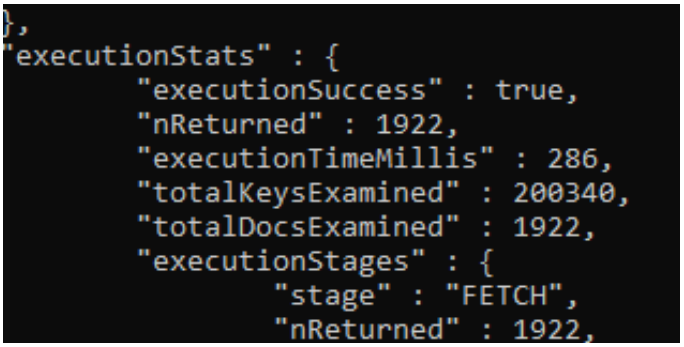
- Facundo Cadaa

  búsqueda sin index

creando index



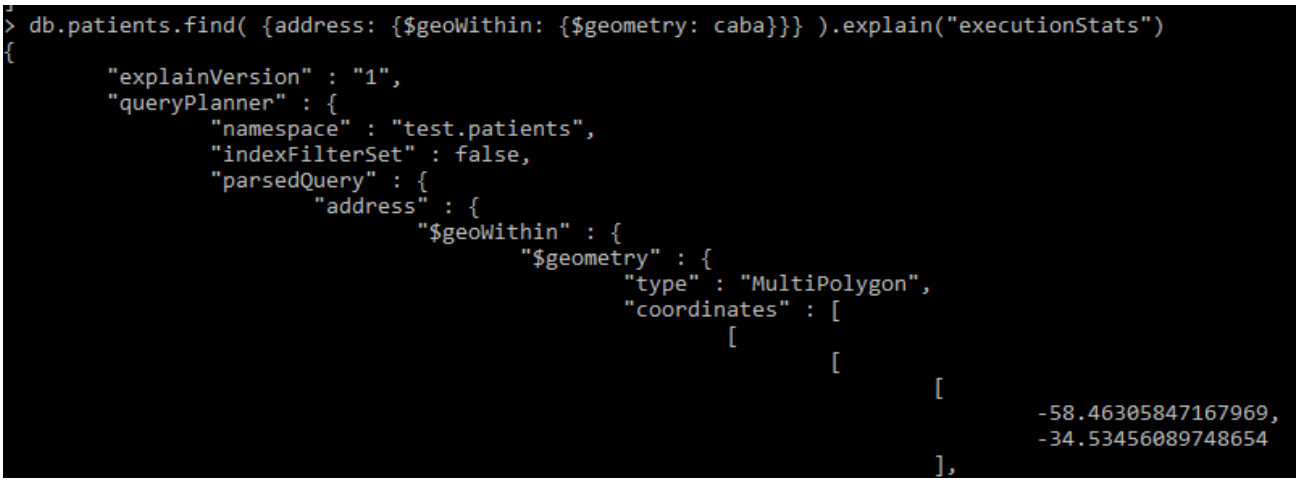
búsqueda con index

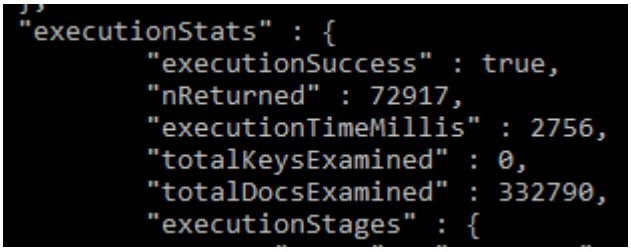


13. Busque los pacientes que viven dentro de la ciudad de Buenos Aires. Para esto, puede definir una variable en la terminal y asignarle como valor el polígono del archivo provisto **`caba.geojson`** (copiando y pegando directamente). Cree un índice geoespacial de tipo **`2dsphere`** para el campo **`location`** de la colección **`patients`** y, de la misma forma que en el punto 12, compare la performance de la consulta con y sin dicho índice.

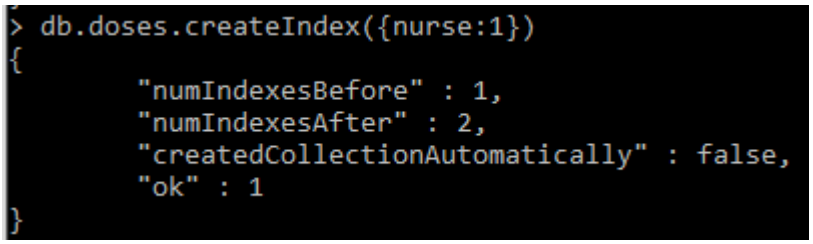
- Franco Camacho

sin índice





con índice



```
> db.patients.find( {address: {$geoWithin: {$geometry: caba}}} ).explain("executionStats")
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "test.patients",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "address" : {
                                "$geoWithin" : {
                                        "$geometry" : {
                                                "type" : "MultiPolygon",
                                                "coordinates" : [
                                                        [
                                                                [
                                                                        [
                                                                                -58.46305847167969,
                                                                                -34.53456089748654
                                                                        ],
```

```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 72917,
        "executionTimeMillis" : 1229,
        "totalKeysExamined" : 91926,
        "totalDocsExamined" : 91906,
        "executionStages" : {
```

- Joaquin Galdeano

```
executionStats:
 { executionSuccess: true,
   nReturned: 43623,
   executionTimeMillis: 414,
   totalKeysExamined: 0,
   totalDocsExamined: 199168,
   executionStages:
```

Sin índices, en la consulta (b) por pacientes de Buenos Aires (db.patients.find({address: {$geoWithin: {$geometry: doc} } }).explain("executionStats")) se examinaron 199168 documentos en 414 milisegundos

Creación de índice por campo address de la colección patients:

```
> db.patients.createIndex( { address : "2dsphere" } )
< 'address_2dsphere'
```

Con el nuevo índice creado, la misma consulta (b) examina 55159 documentos en 293 milisegundos.

```
executionStats:
 { executionSuccess: true,
   nReturned: 43623,
   executionTimeMillis: 293,
   totalKeysExamined: 55178,
   totalDocsExamined: 55159,
   executionStages:
```

- Facundo Cadaa

  búsqueda sin index

```
  },
  "executionStats" : {
          "executionSuccess" : true,
          "nReturned" : 43927,
          "executionTimeMillis" : 686,
          "totalKeysExamined" : 0,
          "totalDocsExamined" : 200340,
          "executionStages" : {
                  "stage" : "COLLSCAN",
```

  creando index

```
> db.patients.createIndex( { address : "2dsphere" } )
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
>
```

  búsqueda con index

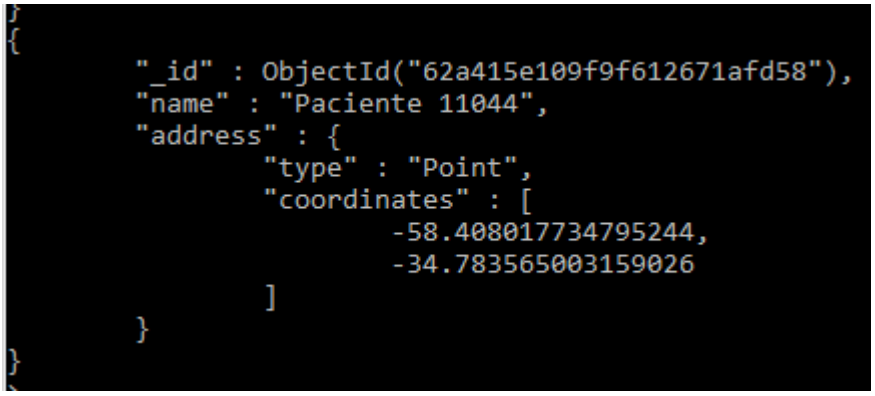
```
},
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 43927,
        "executionTimeMillis" : 377,
        "totalKeysExamined" : 55559,
        "totalDocsExamined" : 55540,
        "executionStages" : {
                "stage" : "FETCH"
```

Parte 4: Aggregation Framework

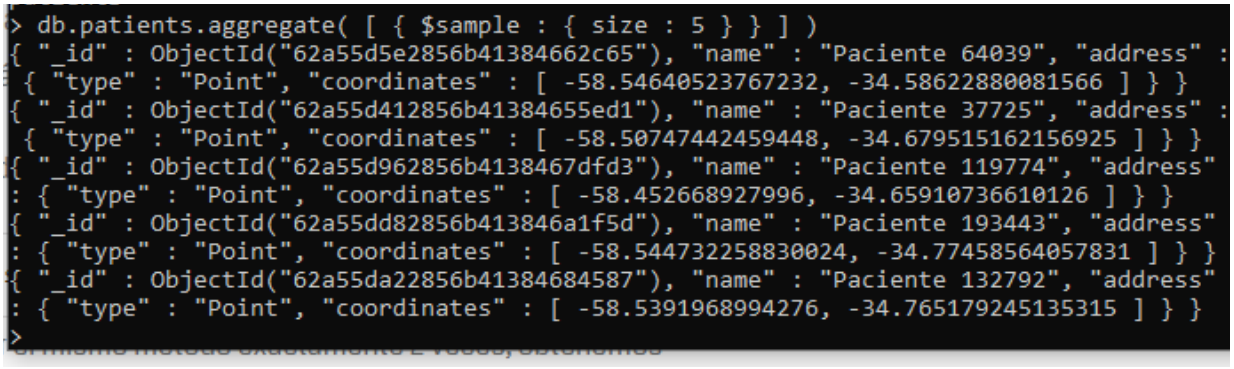
▸MongoDB cuenta con un Aggregation Framework que brinda la posibilidad de hacer analítica en tiempo real del estilo OLAP (Online Analytical Processing), de forma similar a otros productos específicos como Hadoop o MapReduce. En los siguientes ejercicios se verán algunos ejemplos de su aplicabilidad.

14. Obtenga 5 pacientes aleatorios de la colección

Franco Camacho

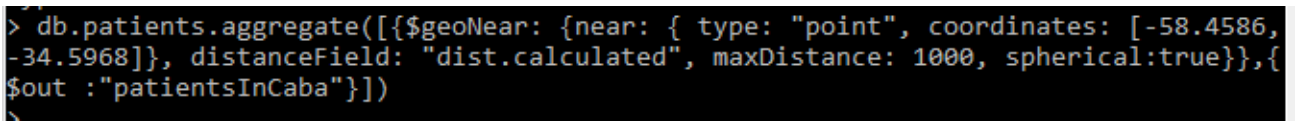


Facundo Cadaa

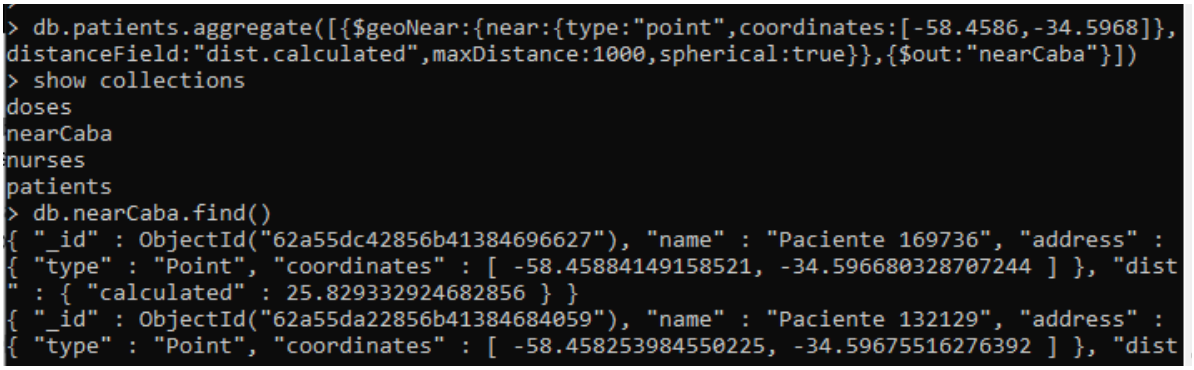


15. Usando el framework de agregación, obtenga los pacientes que vivan a 1km (o menos) del centro geográfico de la ciudad de Buenos Aires (**`[-58.4586,-34.5968]`**) y guárdelos en una nueva colección.

Franco Camacho



Facundo Cadaa



16. Obtenga una colección de las dosis aplicadas a los pacientes del punto anterior. Note que sólo es posible ligarlas por el nombre del paciente.

```
db.patientsInCaba.aggregate([{
 $lookup: {
  from: 'doses',
  localField: 'name',
  foreignField: 'patient',
  as: 'dosesNearCaba'
 }
}, {
 $project: {
  _id: 1,
  name: 1,
  dosesNearCaba: 1
 }
}, {
 $unwind: {
  path: '$dosesNearCaba'
 }
}, {
 $replaceRoot: {
  newRoot: '$dosesNearCaba'
 }
},{$out:"appliedDose"}])
```

▸ Si la consulta se empieza a tornar difícil de leer, se pueden ir guardando los agregadores en variables, que no son más que objetos en formato JSON.

17. Obtenga una nueva colección de *nurses*, cuyos nombres incluyan el string “111”. En cada documento (cada *nurse*) se debe agregar un atributo *doses* que consista en un array con todas las dosis que aplicó después del 1/5/2021.

```
db.nurses.aggregate([{ $match: { name: { $regex: '111' } } }, { $addFields: { doses: [] } }, { $lookup: { from: 'doses', localField: 'name', foreignField: 'nurse', as: 'doses', pipeline: [ { $match: { date: { $gt: ISODate('2021-05-01') } } } ] } }, { $out: 'nurses_punto17' }])
```

(1) $match busca los enfermeros de *nurses* con el string “111” en su nombre.
(2) $addFields le agrega a los documentos encontrados un array vacío llamado “doses”.
(3) $lookup hace un join entre la colección *nurses* y *doses* a través de los campos *name* y *nurse* de cada uno respectivamente y al resultado lo guarda en *doses*. En el pipeline del $lookup se hace un (4) $match para filtrar *doses* por el campo *date* mientras *date* sea mayor a 1/5/2021, y al resultado lo guarda a través de (5) $out en una nueva colección llamada ‘nurses_punto17’