
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
    - MongoDB es una **base de datos** orientada a documentos que ofrece una gran escalabilidad y flexibilidad y un modelo de consultas e indexación avanzado. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.
    - Una de las diferencias más importantes con respecto a las **bases de datos relacionales**, es que no es necesario seguir un esquema. Carece de lenguaje DDL.
- 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.

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

En la versión MongoDB 4.0 se implementa un modelo transaccional ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) sobre diferentes documentos.
Hasta la aparición de este modelo transaccional, MongoDB trabajaba de forma atómica sobre cada uno de los documentos, ofreciéndoles a ellos el modelo transaccional. En esta situación la forma de modelar las referencias entre documentos para poder asegurar una integridad se realizaban mediante anidación de documentos. Este modelo para la gestión de transacciones sigue siendo válido, aunque dispongamos del nuevo mecanismo ACID.

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

Los índices se definen a nivel de las colecciones, se soportan índices por cualquier campo o subcampo del documento. Los tipos que hay son:

- Single Field: simple, un único campo.
- Compound Index: compuesto, más de un campo.
- Multikey index: más de una clave.
- Geospatial index: utilizan datos tanto en formato de coordenadas como GeoJSON.
- Text index: una colección puede tener a lo sumo un índice orientado a texto. Permite hacer búsquedas por partes de texto, como cadena de caracteres incompletas, de forma más eficiente.
- Hashed index

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 referenciarlo y obtenerlo a través de una segunda query.
- DBRefs: se utiliza cuando un documento contiene referencias a otro documento presente en diferentes colecciones.

Parte 2: Primeros pasos con MongoDB

5. Cree una nueva base de datos llamada airbdb, y una colección llamada apartments. En esa colección inserte un nuevo documento (un departamento) con los siguientes atributos:

{name:'Apartment with 2 bedrooms', capacity:4}

recupere la información del departamento usando el comando db.apartments.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?

Se crea la base de datos:

```
> use airbdb
switched to db airbdb
```

Se crea la colección y se inserta el registro:

```
> db.apartments.insert({name:'Apartment with 2 bedrooms', capacity:4})
WriteResult({ "nInserted" : 1 })
>
```

Se recupera la información:

```
> db.apartments.find().pretty()
{
        "_id" : ObjectId("5eb3629ee26cca18b76069e7"),
        "name" : "Apartment with 2 bedrooms",
        "capacity" : 4
}
>
```

La única diferencia es que además de los datos insertados, nos devuelve un dato **_id,** el cual no formaba parte de la inserción. Este **_id** es un valor/atributo que genera automáticamente Mongo como identificador de cada documento, con el fin de poder identificar unívocamente cada documento dentro de cada colección dentro de la base de datos.

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

- {name:'New Apartment', capacity:3, services: ['wifi', 'ac']}
- {name:'Nice apt for 6', capacity:6, services: ['parking']}
- {name:'1950s Apartment', capacity:3}
- {name:'Duplex Floor', capacity:4, services: ['wifi', 'breakfast', 'laundry']}
-

```
> db.apartments.insertMany(
... [ {name:'Apartment with 2 bedrooms', capacity:4}, {name:'New Apartment', capacity:3, serv
ices: ['wifi', 'ac']}, {name:'Nice apt for 6', capacity:6, services: ['parking']}, {name:'195
0s Apartment', capacity:3}, {name:'Duplex Floor', capacity:4, services: ['wifi', 'breakfast',
 'laundry']} ]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5eb36e7fceaf2cac2ac7a6da"),
                ObjectId("5eb36e7fceaf2cac2ac7a6db"),
                ObjectId("5eb36e7fceaf2cac2ac7a6dc"),
                ObjectId("5eb36e7fceaf2cac2ac7a6dd"),
                ObjectId("5eb36e7fceaf2cac2ac7a6de")
        ]
}
```

Y busque los departamentos:

- con capacidad para 3 personas:

```
> db.apartments.find({ capacity:3 }).pretty();
{
        "_id" : ObjectId("5eb3415fd11c62757edff174"),
        "name" : "New Apartment",
        "capacity" : 3,
        "services" : [
                "wifi",
                "ac"
        ]
}
{
        "_id" : ObjectId("5eb341e4d11c62757edff176"),
        "name" : "1950s Apartment",
        "capacity" : 3
}
>
```

- con capacidad para 4 personas o más:

```
> db.apartments.find({capacity: { $gt: 4 } }).pretty();
{
        "_id" : ObjectId("5eb36794d11c62757edff17a"),
        "name" : "Nice apt for 6",
        "capacity" : 6,
        "services" : [
                "parking"
        ]
}
```

- con wifi:

```
> db.apartments.find({"services": "wifi"}).pretty()
{
        "_id" : ObjectId("5eb3677cd11c62757edff179"),
        "name" : "New Apartment",
        "capacity" : 3,
        "services" : [
                "wifi",
                "ac"
        ]
}
{
        "_id" : ObjectId("5eb367cbd11c62757edff17c"),
        "name" : "Duplex Floor",
        "capacity" : 4,
        "services" : [
                "wifi",
                "breakfast",
                "laundry"
        ]
```

- que incluyan la palabra ‘Apartment’ en su nombre:

```
> db.apartments.find({"name": /.*Apartment.*/}).pretty()
{
        "_id" : ObjectId("5eb3675fd11c62757edff178"),
        "name" : "Apartment with 2 bedrooms",
        "capacity" : 4
}
{
        "_id" : ObjectId("5eb3677cd11c62757edff179"),
        "name" : "New Apartment",
        "capacity" : 3,
        "services" : [
                "wifi",
                "ac"
        ]
}
{
        "_id" : ObjectId("5eb367a8d11c62757edff17b"),
        "name" : "1950s Apartment",
        "capacity" : 3
}
```

- con la palabra ‘Apartment’ en su nombre y capacidad para más de 3 personas:

```
> db.apartments.find({ $and: [{"name": /.*Apartment.*/}, {"capacity": {$gt:3}}] }).pretty()
{
        "_id" : ObjectId("5eb3675fd11c62757edff178"),
        "name" : "Apartment with 2 bedrooms",
        "capacity" : 4
}
```

- sin servicios (es decir, que el atributo esté ausente):

```
> db.apartments.find({"services": { $exists: false } }).pretty()
{
        "_id" : ObjectId("5eb3675fd11c62757edff178"),
        "name" : "Apartment with 2 bedrooms",
        "capacity" : 4
}
{
        "_id" : ObjectId("5eb367a8d11c62757edff17b"),
        "name" : "1950s Apartment",
        "capacity" : 3
}
```

```
> db.apartments.find({"services": {$eq: null}}).pretty()
{
        "_id" : ObjectId("5eb3629ee26cca18b76069e7"),
        "name" : "Apartment with 2 bedrooms",
        "capacity" : 4
}
{
        "_id" : ObjectId("5eb36536e26cca18b76069ea"),
        "name" : "1950s Apartment",
        "capacity" : 3
}
```

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

```
> db.apartments.find({services: {$exists: false}}, {_id:0, name:1}).pretty()
{ "name" : "Apartment with 2 bedrooms" }
{ "name" : "1950s Apartment" }
>
```

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

También se pueden hacer con findAndModify() en lugar de Update.

7. Actualice el “Duplex Floor” asignándole capacidad 5.

```
> db.apartments.updateOne({"name": "Duplex Floor"},{$set: {"capacity": 5}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 0 }
```

```
> db.apartments.find({"name": "Duplex Floor"}).pretty()
{
        "_id" : ObjectId("5eb36550e26cca18b76069eb"),
        "name" : "Duplex Floor",
        "capacity" : 5,
        "services" : [
                "wifi",
                "breakfast",
                "laundry"
        ]
}
>
```

8. Agregue “laundry” al listado de services del “Nice apt for 6”.

```
>
>
> db.apartments.updateOne({"name": "Nice apt for 6"},{$addToSet: { services : "laundry"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
>
>
>
> db.apartments.find({"name": "Nice apt for 6"}).pretty()
{
        "_id" : ObjectId("5eb36526e26cca18b76069e9"),
        "name" : "Nice apt for 6",
        "capacity" : 6,
        "services" : [
                "parking",
                "laundry"
        ]
}
>
```

9. Agregue una persona más de capacidad a todos los departamentos con wifi.

```
>
>
> db.apartments.find({"services": "wifi"}).pretty()
{
        "_id" : ObjectId("5eb36512e26cca18b76069e8"),
        "name" : "New Apartment",
        "capacity" : 3,
        "services" : [
                "wifi",
                "ac"
        ]
}
{
        "_id" : ObjectId("5eb36550e26cca18b76069eb"),
        "name" : "Duplex Floor",
        "capacity" : 5,
        "services" : [
                "wifi",
                "breakfast",
                "laundry"
        ]
}
```

```
>
>
> db.apartments.update({"services": "wifi"}, { $inc: { capacity: 1 } },{multi:true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
>
>
```

```
>
> db.apartments.find({"services": "wifi"}).pretty()
{
        "_id" : ObjectId("5eb36512e26cca18b76069e8"),
        "name" : "New Apartment",
        "capacity" : 4,
        "services" : [
                "wifi",
                "ac"
        ]
}
{
        "_id" : ObjectId("5eb36550e26cca18b76069eb"),
        "name" : "Duplex Floor",
        "capacity" : 6,
        "services" : [
                "wifi",
                "breakfast",
                "laundry"
        ]
}
>
>
```

Parte 3: Índices

▸ Elimine todos los departamentos de la colección. Guarde en un archivo llamado 'generador.js' el siguiente código JavaScript y ejecútelo con: load().

```
for (var i = 1; i <= 50000; i++) {
      var randomServices = ['wifi', 'pool', 'parking', 'breakfast'].sort( function()
{ return 0.5 - Math.random() } ).slice(1, Math.floor(Math.random() * 5));
      var randomCapacity = Math.ceil(Math.random() * 5);
      var randomLong = ((Math.random()/1.3)+51);
      var randomLat = Math.random() - .4;
      db.apartments.insert({
            name:'Apartment '+i,
            capacity:randomCapacity,
            services: randomServices,
            location: {
           type: "Point",
           coordinates: [randomLat, randomLong]
         }
      });
}
```

Con drop eliminás la colección:

```
>
> db.apartments.drop()
true
```

Con remove vacías la colección:

```
> db.apartments.remove({})
WriteResult({ "nRemoved" : 5 })
```

```
> load("/home/yani/Facultad/2020/BBDD2/generador.js")
true
```

10. Busque en la colección de departamentos si existe algún índice definido.

MongoDB crea un índice sobre el campo _id durante la creación de una colección. Este índice controla la unicidad del valor del _id y no puede ser eliminado.

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

11. Cree un índice para el campo name. Busque los departamentos que tengan en su nombre 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.

```
}
> db.apartments.createIndex( { name: 1 }, { collation: { locale: "es" } } )
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
>
```

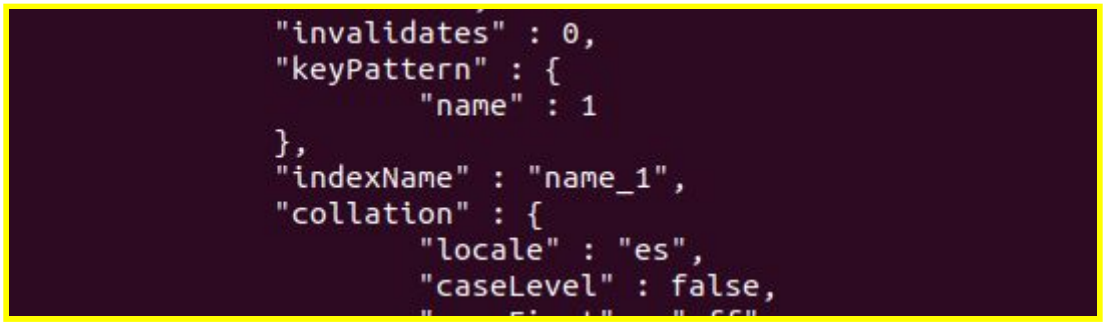
```
}
> db.apartments.find({name: /.*11.*/}).collation({locale:"es"}).explain("executionStats")
{
```

Sin índice:

```
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 2291,
                "executionTimeMillis" : 50,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 50000,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "name" : {
                                        "$regex" : ".*11.*"
                                }
                        },
```

Con índice:

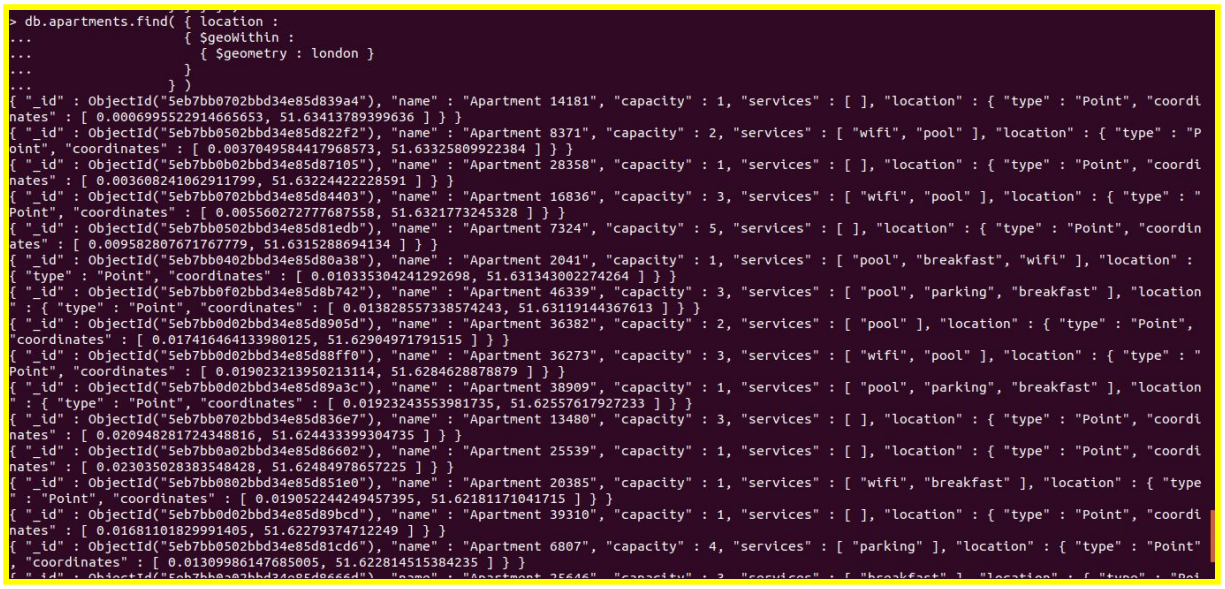
```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 2291,
        "executionTimeMillis" : 71,
        "totalKeysExamined" : 50000,
        "totalDocsExamined" : 50000,
        "executionStages" : {
                "stage" : "FETCH",
                "filter" : {
                        "name" : {
                                "$regex" : ".*11.*"
                        }
                },
```

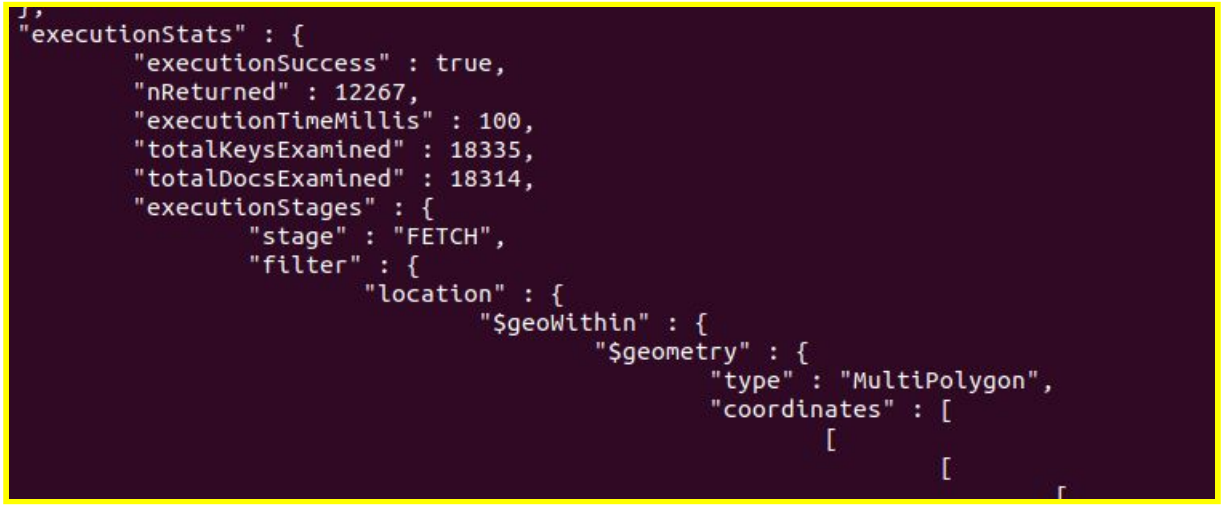
```
                "invalidates" : 0,
                "keyPattern" : {
                        "name" : 1
                },
                "indexName" : "name_1",
                "collation" : {
                        "locale" : "es",
                        "caseLevel" : false,
```


12. Busque los departamentos dentro de la ciudad de Londres. Para esto, puede definir una variable en la terminal y asignarle como valor el polígono del archivo provisto greaterlondon.geojson (copiando y pegando directamente). Cree un índice geoespacial de tipo 2dsphere para el campo location de la colección apartments y, de la misma forma que en el punto 11, compare la performance de la consulta con y sin dicho índice.

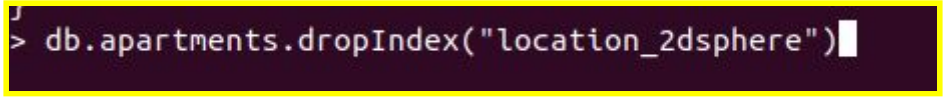
```
> db.apartments.createIndex({location:'2dsphere'});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.apartments.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.apartments"
        },
        {
                "v" : 2,
                "key" : {
                        "location" : "2dsphere"
                },
```

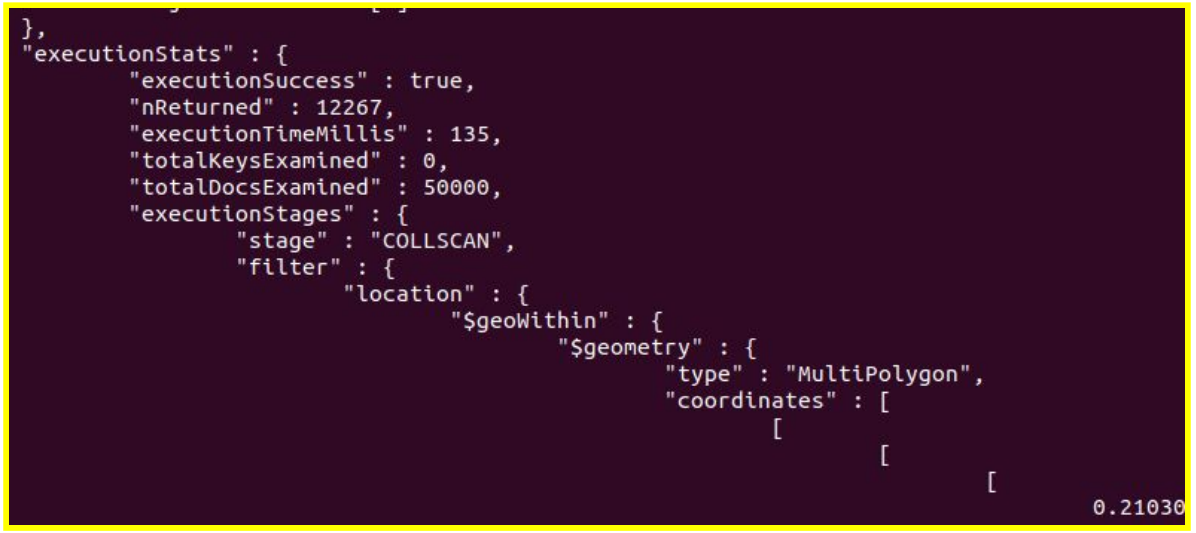


Con índice:



Sin índice:





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.
Al igual que en la parte 3, guarde en un archivo llamado 'generadorReservas.js' el siguiente código JavaScript y ejecútelo con la función load():

```
Date.prototype.addDays=function(d){return new Date(this.valueOf()+864E5*d);};
function randomDate(start, end) {
  return new Date(start.getTime()+Math.random()*(end.getTime()-start.getTime()));
}
for (var i = 1; i <= 50000; i++) {
      if (Math.random() > 0.7) {
            var randomReservations = Math.ceil(Math.random() * 5);
            for (var r = 1; r <= randomReservations; r++){
                  var startDate = randomDate(new Date(2012, 0, 1), new Date());
                  var days = Math.ceil(Math.random()*8);
                  var toDate = startDate.addDays(days);
                  var randomAmount = days * ((Math.random() * 100) + 80).toFixed(2);
                  db.reservations.insert({
                        apartmentName:'Apartment '+i,
                        from: startDate,
                        to: toDate,
                        amount: randomAmount
                  });}
      }
}
```

```
}
> load("/home/yani/Facultad/2020/BBDD2/generadorReservas.js")
true
>
```

13. Obtenga 5 departamentos aleatorios de la colección.

```
> db.apartments.aggregate(
...     [ { $sample: { size: 5 } } ]
... )
{ "_id" : ObjectId("5eb7bb0302bbd34e85d80240"), "name" : "Apartment 1", "capacit
y" : 3, "services" : [ ], "location" : { "type" : "Point", "coordinates" : [ 0.3
7810625202540027, 51.53982966181989 ] } }
{ "_id" : ObjectId("5eb7bb1002bbd34e85d8c469"), "name" : "Apartment 49706", "cap
acity" : 4, "services" : [ ], "location" : { "type" : "Point", "coordinates" : [
 -0.34894399204460236, 51.12775632011733 ] } }
{ "_id" : ObjectId("5eb7bb0e02bbd34e85d8ac17"), "name" : "Apartment 43480", "cap
acity" : 2, "services" : [ ], "location" : { "type" : "Point", "coordinates" : [
 -0.3128880950379489, 51.715161413588866 ] } }
{ "_id" : ObjectId("5eb7bb0f02bbd34e85d8bbf4"), "name" : "Apartment 47541", "cap
acity" : 4, "services" : [ "pool", "parking" ], "location" : { "type" : "Point",
 "coordinates" : [ -0.04132867268884688, 51.347467202034906 ] } }
{ "_id" : ObjectId("5eb7bb0f02bbd34e85d8b609"), "name" : "Apartment 46026", "cap
acity" : 5, "services" : [ ], "location" : { "type" : "Point", "coordinates" : [
 -0.013039589483165748, 51.61766456498628 ] } }
```

14. Usando el framework de agregación, obtenga los departamentos que estén a 15km (o menos) del centro de la ciudad de Londres ([-0.127718, 51.507451]) y guárdelos en una nueva colección.

```
> db.apartments.aggregate([
...     {
...       $geoNear: {
...          near: { type: "Point", coordinates: [-0.127718, 51.507451] },
...          distanceField: "dist.calculated",
...          maxDistance: 1500,
...          spherical: true
...       }
...     },
...     { $out : "apartments15kmLondon" }
... ])
> db.apartments15kmLondon.find().pretty()
{
        "_id" : ObjectId("5eb7bb0902bbd34e85d861cf"),
        "name" : "Apartment 24464",
        "capacity" : 1,
        "services" : [
                "pool"
        ],
        "location" : {
                "type" : "Point",
                "coordinates" : [
                        -0.12494866052087428,
                        51.508095714541014
                ]
        },
        "dist" : {
                "calculated" : 204.85878784179337
        }
}
{
        "_id" : ObjectId("5eb7bb0902bbd34e85d85e86"),
        "name" : "Apartment 23623",
        "capacity" : 4,
        "services" : [ ],
        "location" : {
                "type" : "Point",
                "coordinates" : [
```

15. Para los departamentos hallados en el punto anterior, obtener una colección con cada departamento agregando un atributo reservas que contenga un array con todas sus reservas. Note que sólo es posible ligarlas por el nombre del departamento.

```
> db.apartments15kmLondon.aggregate([
...     {
...       $lookup:
...         {
...           from: "reservations",
...           localField: "name",
...           foreignField: "apartmentName",
...           as: "reservations"
...         }
...    },
...    { $out : "apartmentsWithReservations" }
... ])
```

```
> db.apartmentsWithReservations.find().pretty()
{
        "_id" : ObjectId("5eb7bb0902bbd34e85d861cf"),
        "name" : "Apartment 24464",
        "capacity" : 1,
        "services" : [
                "pool"
        ],
        "location" : {
                "type" : "Point",
                "coordinates" : [
                        -0.12494866052087428,
                        51.508095714541014
                ]
        },
        "dist" : {
                "calculated" : 204.85878784179337
        },
        "reservations" : [ ]
}
{
        "_id" : ObjectId("5eb7bb0902bbd34e85d85e86"),
        "name" : "Apartment 23623",
        "capacity" : 4,
        "services" : [ ],
        "location" : {
                "type" : "Point",
                "coordinates" : [
                        -0.12419249008820477,
                        51.50591811662225
                ]
        },
        "dist" : {
                "calculated" : 297.97178235072556
        },
        "reservations" : [ ]
}
```

```
{
        "_id" : ObjectId("5eb7bb0602bbd34e85d833aa"),
        "name" : "Apartment 12651",
        "capacity" : 2,
        "services" : [
                "wifi"
        ],
        "location" : {
                "type" : "Point",
                "coordinates" : [
                        -0.12297367005765214,
                        51.507824964017274
                ]
        },
        "dist" : {
                "calculated" : 331.3411481270413
        },
        "reservations" : [
                {
                        "_id" : ObjectId("5eb7c56c02bbd34e85d8f21f"),
                        "apartmentName" : "Apartment 12651",
                        "from" : ISODate("2018-12-03T16:09:31.611Z"),
                        "to" : ISODate("2018-12-08T16:09:31.611Z"),
                        "amount" : 506.85
                },
                {
                        "_id" : ObjectId("5eb7c56c02bbd34e85d8f220"),
                        "apartmentName" : "Apartment 12651",
                        "from" : ISODate("2018-06-07T05:38:50.824Z"),
                        "to" : ISODate("2018-06-15T05:38:50.824Z"),
                        "amount" : 797.44
                }
        ]
}
```

▸ 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.

16. Usando la colección del punto anterior, obtenga el promedio de precio pagado por reserva (precio completo, no dividir por la cantidad de noches) de cada departamento.

```
> db.apartmentsWithReservations.aggregate(
...     [
...       { $unwind: "$reservations"},
...       {
...         $group:
...           {
...             _id: { name: "$name" },
...             avg: { $avg: "$reservations.amount" }
...           }
...       }
...     ]
... )
{ "_id" : { "name" : "Apartment 16761" }, "avg" : 1015.74 }
{ "_id" : { "name" : "Apartment 18398" }, "avg" : 430.83666666666664 }
{ "_id" : { "name" : "Apartment 26465" }, "avg" : 417.68199999999996 }
{ "_id" : { "name" : "Apartment 41612" }, "avg" : 871.62 }
{ "_id" : { "name" : "Apartment 12651" }, "avg" : 652.145 }
{ "_id" : { "name" : "Apartment 24399" }, "avg" : 769.2 }
{ "_id" : { "name" : "Apartment 542" }, "avg" : 244.86 }
{ "_id" : { "name" : "Apartment 8262" }, "avg" : 413.524 }
{ "_id" : { "name" : "Apartment 20709" }, "avg" : 466.74 }
{ "_id" : { "name" : "Apartment 35111" }, "avg" : 203.23000000000002 }
{ "_id" : { "name" : "Apartment 9012" }, "avg" : 396.976 }
```