

Tema4- Otros modelos de datos para SI

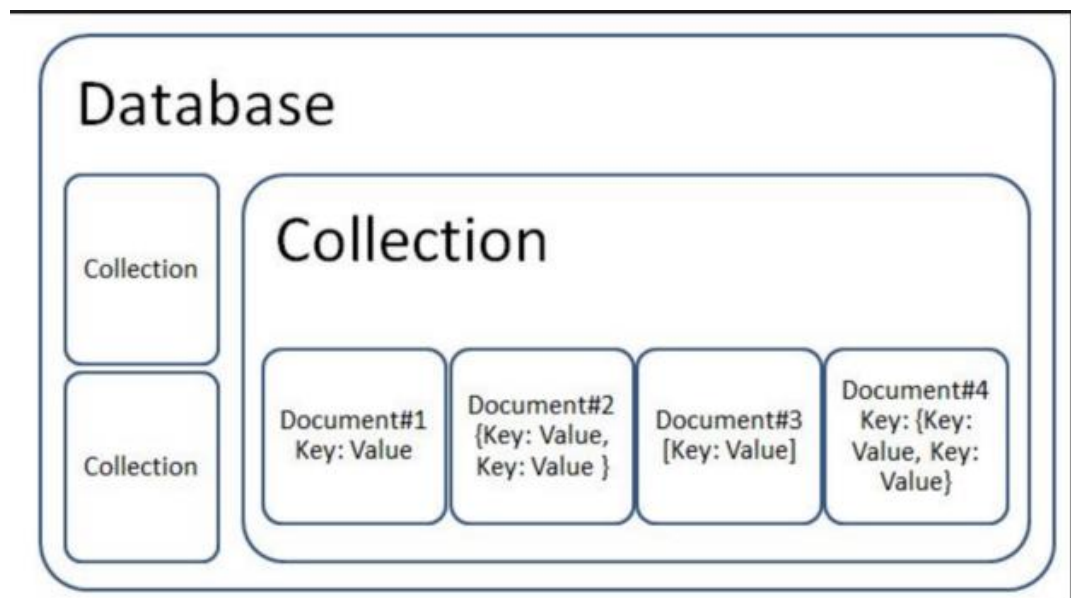
Modelos NoSQL

- **Breve descripción de la descarga e instalación del SGBD**

Para este trabajo nosotros usaremos el SGBD **mongodb** un sistema NoSQL, orientado a documentos de código abierto.

Este SGBD trabaja con BSON (codificación binaria de JSON) con un esquema dinámico.

Estos documentos almacenan un conjunto de pares clave-valor, pudiendo crear estructuras complejas (los valores pueden ser a la vez conjuntos de pares clave-valor).




- Lo primero que debemos hacer será irnos a la página principal de mongodb (<https://www.mongodb.com/es>) y aquí nos dirigiremos a la sección de community server (<https://www.mongodb.com/try/download/community>) y nos descargaremos la última versión.

Available Downloads ^

Version
4.4.3 (current) ✓

Platform
Windows ✓

Package
msi ✓

 **Download** [Copy Link](#)

[Current releases & packages](#)
[Development releases](#)
[Archived releases](#)
[Changelog](#)
[Release Notes](#)



- Ejecutamos el instalador una vez descargado.
- Le damos a “Next” las veces que nos lo solicite asegurándonos de desmarcar la opción “Install MongoDB as a Service” cuando podamos.

Una vez terminamos la instalación pasaremos a ejecutar el programa, nos iremos a donde por defecto se ha instalado o donde tu le hayas indicado al programa que se instale, si decidimos instalar donde nos pone por defecto, deberemos irnos a disco local C, archivos de programa, mongoDB, Server, 4.4 (Nuestra versión), bin, y ya en esta subcarpeta, nos fijaremos en la aplicación mongod.exe y mongo.exe.

(C:\Program Files\MongoDB\Server\4.4\bin)

Bien antes de poder ejecutar nuestra aplicación deberemos de crear una carpeta, en nuestro disco C en la raíz (C:\) de nombre **data**, dentro de esta deberemos crear otra de nombre **db**.

Una vez hecho esto lanzaremos la aplicación de mongod.exe que será nuestra base de datos, lo dejaremos lanzado, mientras ejecutamos mongo.exe nuestra terminal donde pasaremos a trabajar con nuestra base de datos.

● Breve descripción del DDL y DML utilizado

DDL (Data Definition Language) -> Creación de base de datos y todos sus componentes: tablas, índices, relaciones ...

- **use database**
 - “Nos metemos dentro de la base de datos y la crea si no existe.”
- **db**
 - “Nos indica en que base de datos estamos actualmente.”
- **show dbs**
 - “Nos muestra todas las bases de datos que existen.”
- **db.createCollection(“coleccion”)**
 - “Creamos una nueva colección/documento de forma explícita.”
- **show collections**
 - “Muestra todas las colecciones que tenemos en nuestra base de datos.”
- **db.coleccion.insert({“nombre”:“valor”})**
 - “Al hacer un insert sobre una colección que no existe se crea de forma implícita.”
- **db.dropDataBase()**
 - “Para eliminar la base de datos donde nos encontramos ahora mismo.”

DML (Data Manipulation Language) -> Insertar, borrar, modificar y consultar los datos de una base de datos.

- ❖ **db.coleccion.insert({"nombre":"valor"})**
 - Insertamos pares de clave-valor.
- ❖ **db.coleccion.insert([{"nombre":"valor"}, {"nombre2":"valor2"}])**
 - Manera de insertar más de un par a la vez.
- ❖ **db.coleccion.update({"nombre":"valor"}, {\$set: {'valor': 10}})**
 - Con update actualizaremos el valor del par que hemos identificado al principio con su clave en la colección indicada.
- ❖ **db.coleccion.drop()**
 - Borro una colección.
- ❖ **db.coleccion.deleteOne({"nombre":"valor"})**
 - Borra la tupla con el identificador que le hemos pasado, ese par clave-valor.
- ❖ **db.coleccion.find().pretty()**
 - te muestra todos los pares clave-valor que existen en la colección que hemos solicitado.
- ❖ **db.coleccion.find({"nombre":"valor"})**
 - Busca un par que coincide con el valor.

Operacion	Ejemplo
Igualdad	{ "stock": 0 }
Menor Que	{ "valor": { \$lt: 15.0 } }
Menor o Igual Que	{ "valor": { \$lte: 16.0 } }
Mayor Que	{ "valor": { \$gt: 18.0 } }
Mayor o Igual Que	{ "valor": { \$gte: 16.0 } }
No es Igual	{ "valor": { \$ne: 0 } }
AND	{ {key1: value1, key2:value2} }

Operacion	Ejemplo
OR	{ \$or: [{key1: value1}, {key2:value2}] }
AND + OR	{ key1: value1, \$or: [{ key2: { \$lt: value2 }, {key3: value3} }] }

- **Sentencias empleadas para la creación de estructuras, inserción/modificación/borrado de datos, y consultas.**

- **Create DATABASE**

Al empezar lo primero que tendremos que hacer una vez instalada nuestro SGBD NoSQL mongodb, será levantar el servicio, ejecutando el mongo.exe, tras eso deberemos crear nuestra base de datos donde haremos todas las pruebas necesarias, el comando que usaremos será el siguiente.

- use DDSI

Cuando le insertemos datos dentro de esta base de datos aparecerá como creada. Para visualizar las bases de datos existentes realizaremos el comando: show dbs.

- **Create TABLE**

Si queremos crear una nueva colección, podemos hacerlo implícitamente, insertando datos ya en una colección que aún no ha sido creada, esta se crearía ya a la vez que se inserta el dato, de la siguiente manera.

- db.pedido.insert({"cod_pedido": "cp2077","precio": "10"})

También podemos crear de forma explícita de la siguiente manera:

- db.createCollection("pedido")

Para eliminar una colección se usará el comando:

- db.pedido.drop()

Con show collections podremos ver todas nuestras colecciones creadas en nuestra base de datos de DDSI.

- **Insert**

Para insertar datos en nuestra base de datos DDSI en nuestra colección, usaremos el siguiente comando:

- db.pedido.insert({"cod_pedido": "cp2077","precio": "10"})

Si queremos añadir varios sería:

- db.pedido.insert ([{"cod_pedido": "cp2078","precio": "20"}, {"cod_pedido": "cp2079","precio": "30"}])

```
C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
> use ddsi
switched to db ddsi
> db.createCollection("pedido")
{ "ok" : 1 }
> show collections
pedido
> db.pedido.insert({"cod_pedido":"cp2077", "precio":"10"})
WriteResult({ "nInserted" : 1 })
> db.pedido.find().pretty()
{
  "_id" : ObjectId("5ffec7e22dff7992826a376f"),
  "cod_pedido" : "cp2077",
  "precio" : "10"
}
```

```
C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe
> db.pedido.insert([{"cod_pedido":"cp2078", "precio":"20"}, {"cod_pedido":"cp2079", "precio":"30"}])
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 2,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})
> db.pedido.find().pretty()
{
  "_id" : ObjectId("5ffec7e22dff7992826a376f"),
  "cod_pedido" : "cp2077",
  "precio" : "10"
}
{
  "_id" : ObjectId("5ffeca6b2dff7992826a3770"),
  "cod_pedido" : "cp2078",
  "precio" : "20"
}
{
  "_id" : ObjectId("5ffeca6b2dff7992826a3771"),
  "cod_pedido" : "cp2079",
  "precio" : "30"
}
```

- **Delete**

Para eliminar una colección y todos los datos que ella contiene se usará el comando:

- `db.pedido.drop()`

Usamos “show collection” y vemos que ha desaparecido.

```
C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe
> db.createCollection("borrar")
{ "ok" : 1 }
> show collections
borrar
pedido
> db.borrar.drop()
true
> show collections
pedido
>
```

Para eliminar un documento (tupla):

- `db.pedido.deleteOne({"cod_pedido":"cp2077"})`

- **Update**

- `db.pedido.update({"cod_pedido":"cp2077"},{$set: {'precio':15}})`

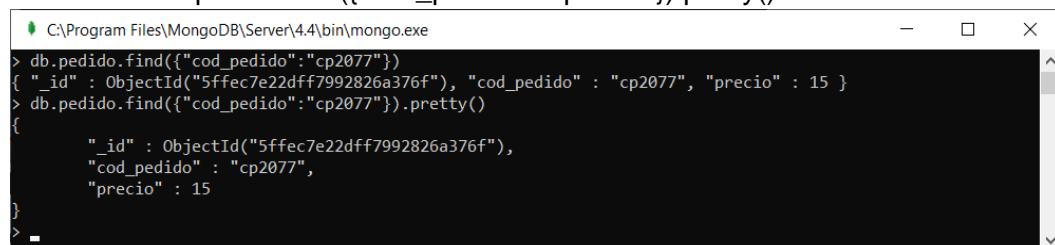


```
C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe
> db.pedido.find().pretty()
{
  "_id" : ObjectId("5ffec7e22dff7992826a376f"),
  "cod_pedido" : "cp2077",
  "precio" : "10"
}
{
  "_id" : ObjectId("5ffeca6b2dff7992826a3770"),
  "cod_pedido" : "cp2078",
  "precio" : "20"
}
{
  "_id" : ObjectId("5ffeca6b2dff7992826a3771"),
  "cod_pedido" : "cp2079",
  "precio" : "30"
}
> db.pedido.update({"cod_pedido":"cp2077"},{$set: {'precio':15}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.pedido.find().pretty()
{
  "_id" : ObjectId("5ffec7e22dff7992826a376f"),
  "cod_pedido" : "cp2077",
  "precio" : 15
}
{
  "_id" : ObjectId("5ffeca6b2dff7992826a3770"),
  "cod_pedido" : "cp2078",
  "precio" : "20"
}
{
  "_id" : ObjectId("5ffeca6b2dff7992826a3771"),
  "cod_pedido" : "cp2079",
  "precio" : "30"
}
>
```

- **Select**

Para pasar a realizar consultas de nuestra base de datos DDSI usamos:

- `db.pedido.find({"cod_pedido":"cp2077"})`
- `db.pedido.find({"cod_pedido":"cp2077"}).pretty()`



```
C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe
> db.pedido.find({"cod_pedido":"cp2077"})
{ "_id" : ObjectId("5ffec7e22dff7992826a376f"), "cod_pedido" : "cp2077", "precio" : 15 }
> db.pedido.find({"cod_pedido":"cp2077"}).pretty()
{
  "_id" : ObjectId("5ffec7e22dff7992826a376f"),
  "cod_pedido" : "cp2077",
  "precio" : 15
}
>
```

- `db.pedido.find().limit(1)`
- `db.pedido.find().sort({'precio':1})` //1 de menor a mayor
// -1 de mayor a menor

```
Seleccionar C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe
> db.pedido.find().limit(1)
{ "_id" : ObjectId("5ffec7e22dff7992826a376f"), "cod_pedido" : "cp2077", "precio" : 15 }
> db.pedido.find().limit(2)
{ "_id" : ObjectId("5ffec7e22dff7992826a376f"), "cod_pedido" : "cp2077", "precio" : 15 }
{ "_id" : ObjectId("5ffeca6b2dff7992826a3770"), "cod_pedido" : "cp2078", "precio" : "20" }
> db.pedido.find().sort({precio:1}).pretty()
{
  "_id" : ObjectId("5ffec7e22dff7992826a376f"),
  "cod_pedido" : "cp2077",
  "precio" : 15
}
{
  "_id" : ObjectId("5ffeca6b2dff7992826a3770"),
  "cod_pedido" : "cp2078",
  "precio" : "20"
}
{
  "_id" : ObjectId("5ffeca6b2dff7992826a3771"),
  "cod_pedido" : "cp2079",
  "precio" : "30"
}
>
```

- Breve descripción del mecanismo de conexión al SGBD desde una aplicación.

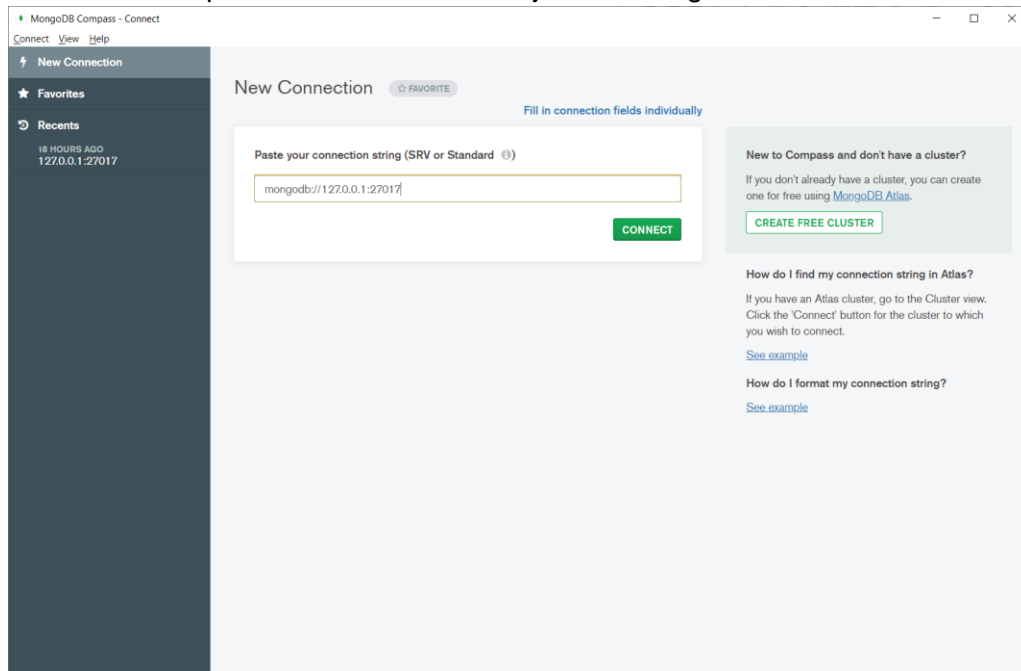
Nos descargamos la aplicación **MongoDB compass**, y nos conectaremos a la nuestra desde esta aplicación.

```
Seleccionar C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe
MongoDB shell version v4.4.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("86efdb82-5983-40d4-9d77-b83d3b04040a") }
MongoDB server version: 4.4.3
---
The server generated these startup warnings when booting:
  2021-01-12T19:36:32.800+01:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
  2021-01-12T19:36:32.801+01:00: This server is bound to localhost. Remote systems will be unable to connect to th
is server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or wi
th --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to di
sable this warning
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

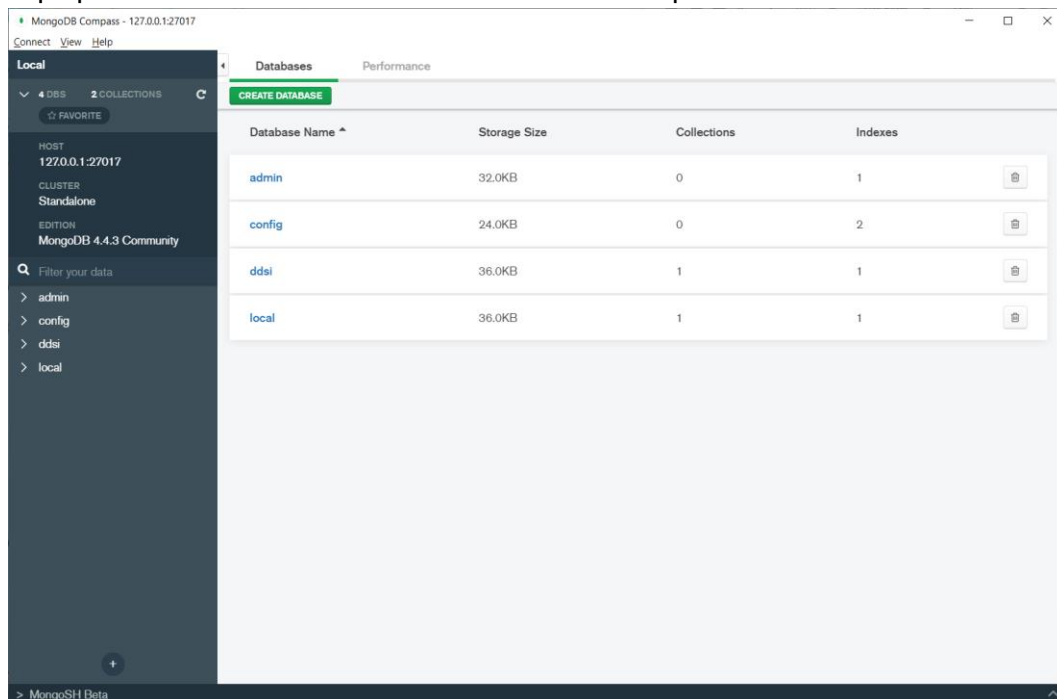
  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> _
```

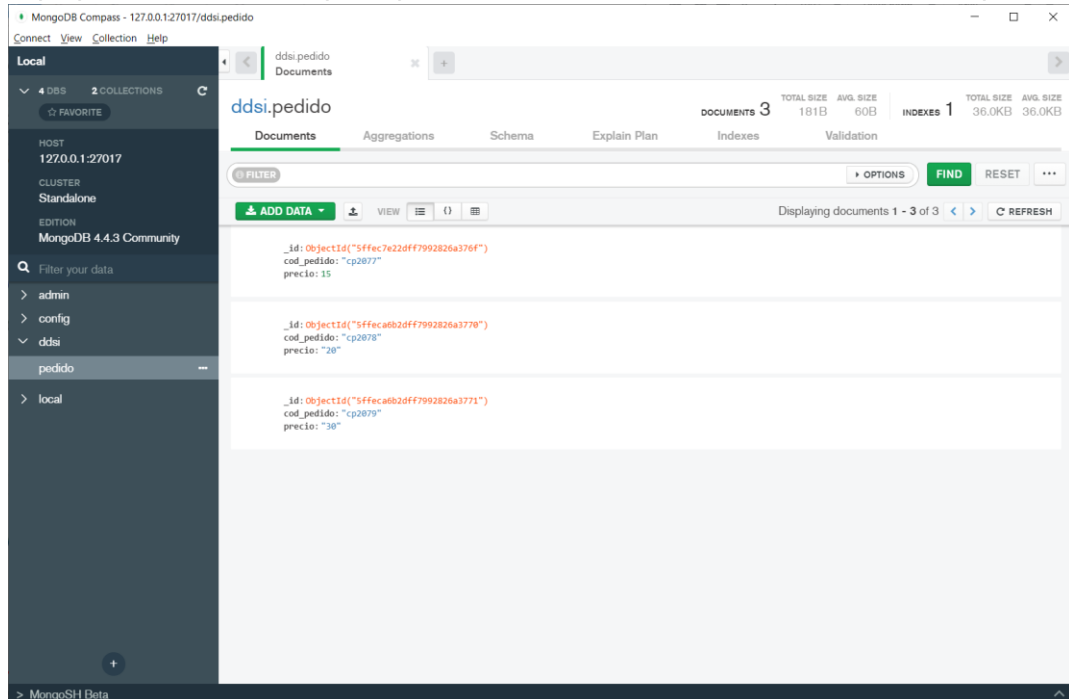

Pondremos los datos que nos encontramos al ejecutar mongo.exe.



Aquí podemos ver todas nuestras bases de datos que tenemos en nuestra conexión.



Aquí podemos ver las tuplas que contiene nuestra colección, en este caso pedido.



- **Breve discusión sobre si sería adecuado para implementar el SI de la práctica**

Para discutir sobre si sería adecuado para implementar el SI de nuestra base de datos de Mercadona, deberemos ver cuáles son las ventajas de implementar esta misma sobre una base de datos NoSQL.

Para ello sabemos que este tipo de SGBD no relacionales se usan para conseguir un alta eficiencia en el almacenamiento de grandes cantidades de tipos de datos poco compatibles con el modelo relacional, una vez dicho esto y conocidas sus ventajas en flexibilidad, escalabilidad, alto rendimiento y alta funcionalidad podemos ver que para nuestro modelo concreto es mejor una base de datos relacional debido a que no requerimos para la práctica un alto rendimiento, pues solo la usamos nosotros (quizás si la queremos extrapolar a un ámbito donde se usa a diario sí) y no se requiere de una alta Escalabilidad pues como he dicho antes es una base de datos de prueba donde hemos insertado unos cuantos datos a modo de prueba.

Además, nosotros para la base de datos de Mercadona preferimos que se cumplan las transacciones ACID ya que es imprescindible para la gestión de la empresa.

- **Bibliografía**

- <https://www.youtube.com/watch?v=c8n6JsQuX2A>
- <https://www.youtube.com/watch?v=-GLMGXkXa7k>
- <https://www.mongodb.com/es>
- <https://www.mongodb.com/try/download/compass>
- <https://www.mongodb.com/try/download/community>
- <https://medium.com/@mark.rethana/introduction-to-nosql-databases-c5b43f3ca1cc>
- <https://pandorafms.com/blog/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/>
- <https://docs.mongodb.com/manual/reference/method/db.collection.insert/>
- <https://datos.codeandcoke.com/apuntes:mongodb>