MongoDB

Ambrosio Cardoso Jiménez

ambrosio.cj@voaxaca.tecnm.mx

Última revisión: Septiembre 2023

MongoDB

```
{
"_id": 1,
"student_name": "Jasmin Scott",
  "school": {
    "school_id": 226,
    "name": "Tech Secondary",
    "address": "100 Broadway St",
    "city": "New York",
    "state": "NY",
    "zipcode": "10001"
  },
"marks": [98, 93, 95, 88, 100],
}
```

```
mongo
> db.students.find({“student_name”:
  “Jasmin Scott”})
```

SQL

students

id	name	school_id
1	Jasmin Scott	226
...	...	...

marks

id	student_id	mark
10	1	98
...	...	...

school_details

id	name	address	city	state	zipcode
226	Tech Secondary	100 Broadway St	New York	NY	10001
...	...	...	...	...	...

Results

name	mark	school_name	city
Jasmin Scott	98	Tech Secondary	New York
...	...	...	...

```
sql
SELECT s.name, m.mark, d.name as "school name",
d.city
FROM students s
INNER JOIN marks m ON s.id = m.student_id
INNER JOIN school_details d ON s.school_id = d.id
WHERE s.name = "Jasmin Scott";
```

fuente: https://www.mongodb.com/docs/

¿Qué es MongoDB?

MongoDB es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado.

MongoDB server Disponible en: https://www.mongodb.com/try/download/community

Mongo Compass Disponible en: https://www.mongodb.com/try/download/compass

Descripción

En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON que significa Binary JSON o JSON Binario) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fáciles y rápidas.

Características de MongoDB

- Consultas ad hoc
- Indexación
- Replicación
- Balanceo de carga
- Almacenamiento de archivos
- Ejecución de JavaScript del lado del servidor

Características de MongoDB

- **Consultas ad hoc**. Con MongoDb podemos realizar todo tipo de consultas. Podemos hacer búsqueda por campos, consultas de rangos y expresiones regulares. Además, estas consultas pueden devolver un campo específico del documento, pero también puede ser una función JavaScript definida por el usuario.

Características de MongoDB

- **Indexación.** El concepto de índices en MongoDB es similar al empleado en bases de datos relacionales, con la diferencia de que cualquier campo documentado puede ser indexado y añadir múltiples índices secundarios.

Características de MongoDB

- **Replicación.** Del mismo modo, la replicación es un proceso básico en la gestión de bases de datos. MongoDB soporta el tipo de replicación primario-secundario. De este modo, mientras podemos realizar consultas con el primario, el secundario actúa como réplica de datos en solo lectura a modo copia de seguridad con la particularidad de que los nodos secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

Características de MongoDB

- **Balanceo de carga.** **Resulta muy interesante cómo MongoDB puede escalar la carga de trabajo. MongoDB tiene la capacidad de ejecutarse de manera simultánea en múltiples servidores, ofreciendo un balanceo de carga o servicio de replicación de datos, de modo que podemos mantener el sistema funcionando en caso de un fallo del hardware.**

Características de MongoDB

- **Almacenamiento de archivos.** Aprovechando la capacidad de MongoDB para el balanceo de carga y la replicación de datos, Mongo puede ser utilizado también como un sistema de archivos. Esta funcionalidad, llamada GridFS e incluida en la distribución oficial, permite manipular archivos y contenido.

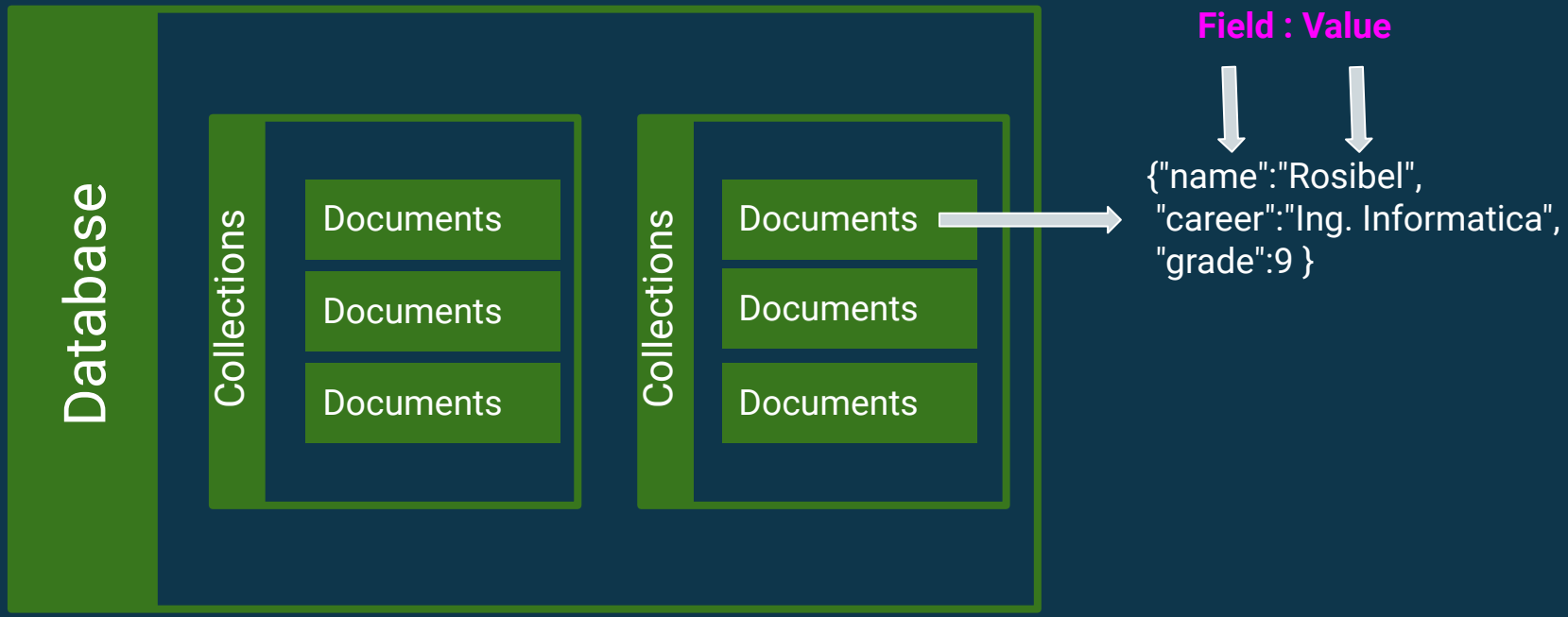
Características de MongoDB

- **Ejecución de JavaScript del lado del servidor.** **MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.**

Terminología

SQL	MongoDB	DynamoDB	Cassandra	CouchBase
Tabla	**Colecciones**	Tabla	Tabla	Bucket de datos
Fila	**Documento**	Elemento	Fila	Documento
Columna	**Campo**	Atributo	Columna	Campo
Clave principal	**ObjectId**	Clave principal	Clave principal	Id del documento

Estructura MongoDB



Mongo en Docker

Para las prácticas de esta unidad instalamos mongo en docker:

Paso 1. docker pull mongo:latest

Paso 2:

```
docker run --name mongo -it --network tbd  -p 27017:27017  \
 -e MONGO_INITDB_ROOT_USERNAME=root \
 -e MONGO_INITDB_ROOT_PASSWORD=t0ps3cr3t \
 -e MONGO_INITDB_DATABASE=testdb \
 -v mongo-data:/data/db \
 -d mongo:latest
```

Mongo shell

MongoDB Shell es una interfaz utilizada en MongoDB que permite a los usuarios interactuar con la base de datos MongoDB para realizar consultas relacionadas con la base de datos para realizar varias operaciones CRUD (Crear, Leer, Actualizar y Eliminar), así como la administración de MongoDB.

Mongo shell

Cuando se instala el servidor MongoDB en la máquina, el programa instalador de MongoDB instala automáticamente MongoDB Shell. También está disponible como programa independiente y puede instalarlo por separado desde el sitio web oficial de MongoDB Inc.

https://www.mongodb.com/try/download/shell

Conexión a mongoDB

Cuando se instala el servidor MongoDB en la máquina, el programa instalador de MongoDB instala automáticamente MongoDB Shell. También está disponible como programa independiente y puede instalarlo por separado desde el sitio web oficial de MongoDB

Mongo shell desde Docker

Entramos al contenedor de esta manera:

docker exec -it mongo bash

Entramos a mongo a través de mongo shell

mongosh --username "root" --password --host localhost --port 27017 --authenticationMechanism SCRAM-SHA-256

Hay que proporcionar el password: ************************

Mongo shell

```
root@22ad8c516680:/# mongosh --username "root" --password --host localhost --port 27017  --authentication
Mechanism SCRAM-SHA-256
Enter password: *********
Current Mongosh Log ID: 6312986021163dd0fa71127f
Connecting to:          mongodb://<credentials>@localhost:27017/?directConnection=true&serverSelectionTim
eoutMS=2000&authMechanism=SCRAM-SHA-256&appName=mongosh+1.5.4
Using MongoDB:          5.0.10
Using Mongosh:          1.5.4

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2022-09-02T23:43:55.477+00:00: Soft rlimits for open file descriptors too low
------

------
   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()
------

test>
```

Crear usuario

La creación de un usuario administrador en MongoDB se realiza mediante el método createUser. El siguiente ejemplo muestra cómo se puede hacer esto.

```
use admin

db.createUser(
{     user: "itvoDeveloper",
      pwd: "t0ps3cr3t",
      roles:[{role: "userAdminAnyDatabase" , db:"admin"},
              {role: "readWrite" , db:"school"},
             ]
})
```

Crear usuario

Crear un usuario para una base de datos específica

```
db.createUser(
{
      user: "cardoso",
      pwd: "t0ps3cr3t",
      roles:[{role: "userAdmin" , db:"books"}]
})
```

Crear usuario

Crear un usuario con diferentes roles en diferentes bases de datos

```
db.createUser(
{
      user: "acardosojmz",
      pwd: "t0ps3cr3t",
      roles:[
            {role: "read" , db:"sales"},
            {role: "readWrite" , db:"school"}
      ]
})
```

Iniciar sesión con el nuevo usuario

```
mongosh --username "itvoDeveloper" --password
--host localhost --port 27017
--authenticationMechanism SCRAM-SHA-256
```

Comandos básicos

Ver las bases de datos actuales: **show dbs**
Ver base de datos actual: **db**
Obtener ayuda: **help**
Crear una nueva base de datos: **use**
Crear una base de datos nueva: **use scholler**
Eliminar una base de datos: **db.dropDatabase()**
Crear una Colección: **db.createCollection("student")**
Ver colecciones: **show collections**
Eliminar colección: **db.student.drop()**

Insertar datos en una colección

```
db.student.insertOne(
{"fullName":"Rosibel Sanchez Lopez","career":"Ing. Informatica", "genre":"M",
"grade":8,"email":["rosibel.sl@voaxaca.tecnm","rosy@gmail.com"]}
)
```

Insertar varios documentos a la vez en una colección

```
db.student.insertMany([
{"fullName":"Jesus Carrasco Ruiz","career":"Ing. Informatica", "genre":"H",
"grade":8,"email":["l17920316@voaxaca.tecnm.mx","jesus@gmail.com"]},
{"fullName":"Jennifer Diego Vásquez","career":"Ing. Informatica", "genre":"M",
"grade":8,"email":["l18920019@voaxaca.tecnm.mx","jeniffer@hotmail.com"]},
{"fullName":"Javier Ivan Aquino Maces","career":"Ing. Informatica",
"genre":"H", "grade":8,"email":["
l18920012@voaxaca.tecnm.mx","javieraquino@hotmail.com"]}
])
```

Recuperar datos de una colección

db.student.find().pretty()

Insertar un nuevo documento con una estructura diferente

```
db.student.insertOne(
{"fullName":"Ana Laura Perez Martinez","career":"Lic. Biologia",
"genre":"M",
"grade":5,"email":["analaura.pm@voaxaca.tecnm","ana@gmail.com"],"
isSingleMother":true}
)
```

Filtrar datos de una colección

Mostrar solamente mujeres

```
db.student.find({genre:"M"})
```

Mostrar estudiantes menores de 8 semestre

```
db.student.find({grade:{$lt:8}})
```

Mostrar estudiantes mayores o iguales 8 semestre

```
db.student.find({grade:{$gte:8}})
```

Filtrar con varias condiciones (AND)

Mostrar mujeres de 5 grado

```
db.student.find({genre:"M", grade:5})
```

Mostrar hombres de 8 grado

```
db.student.find({genre:"H", grade:8})
```

Mostrar estudiantes mayores o iguales 8 semestre

```
db.student.find({grade:{$gte:8}})
```

Filtrar con varias condiciones (OR)

Mostrar mujeres o los de quinto grado

```
db.student.find( { $or: [ { genre: "M" }, { grade: { $eq: 5} } ] } )
```

Mostrar los de Informática o Foresta

```
db.student.find( { $or: [ { career: "Ing. Informatica" },
                          { career: "Ing. Forestal"} ] })
```

Mostrar estudiantes mayores o iguales 8 semestre

```
db.student.find({ $or: [ { career: { $eq: "Ing. Informatica"}},
                         { career: { $eq: "Ing. Forestal"} } ] } )
```

Operaciones de Actualización

```
db.student.updateOne({fullName: "Jesus Carrasco Ruiz"},
                     {$set:{grade: 9}})

db.student.updateMany( { isSingleMother:true }, {$inc: {grade:1}})
```

Operaciones de Eliminación

db.student.insertOne(
{"fullName":"Alumno Temporal","career":"Medicina", "genre":"M",
"grade":1,"email":["temporal@voaxaca.tecnm","temporal@gmail.com"]}
)

db.student.deleteOne({fullName: "Alumno Temporal"})

db.student.deleteMany({ isSingleMother:true })

db.student.deleteMany({}) → Borra todo los documento de la colección

Importar datos desde csv

Abrir

students.csv
~/Documentos

```
"id","fullName","email","career"
20920108,"ALMARAZ MARTINEZ IRIS SAMAHI","L20920108@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920110,"ANTONIO CARLOCK ANGEL FERNANDO","L20920110@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920242,"ARAGON ARAGON ELIANETH","L20920242@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920111,"ARANGO LARA KEVIN JAVIER","L20920111@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
"C17161059","AVENDAÑO SANCHEZ ABILENE GUADALUPE","LC17161059@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920116,"CORTÉS ZÁRATE LETICIA","L20920116@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920118,"CUEVAS MENDOZA GAEL","L20920118@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920119,"ENRIQUEZ CARRILLO DAFNE ESMERALDA","L20920119@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920120,"EUGENIO LOPEZ PATRICIA","L20920120@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920121,"FRANCO JIMENEZ ALEXIS YAHIR","L20920121@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
19920153,"GARCIA PACHECO ALICIA","L19920153@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
"C16161342","JIMENEZ PEREZ ANTONIO DE JESUS","LC16161342@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920129,"JOSE MARTINEZ GENARO","L20920129@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920133,"LÓPEZ LÓPEZ EDUARDO EMILIO","L20920133@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920135,"MENDOZA YESCAS ANDRÉS JAHIR","L20920135@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
"BC15161398","PAZ MARTINEZ MIGUEL ANGEL","LBC15161398@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
19920174,"PEREZ MARISCAL LUIS ANGEL","L19920174@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
"C16161411","REYES MARTINEZ DANIEL","LC16161411@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
20920144,"ROSAS CARRILLO FRANCISCO ANTONIO","L20920144@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
19920182,"SANTAELLA RUIZ FRANCISCO DANIEL","L19920182@voaxaca.tecnm.mx","INGENIERÍA INFORMÁTICA"
```

Importar datos desde csv

Paso 1. Copiar el archivo al contenedor

```
docker cp /home/cardoso/Documentos/students.csv mongo:/
```

Paso 2. Ejecutar la siguiente instrucción desde el contendor sin abrir mongo shell

```
mongoimport --authenticationDatabase admin --username itvoDeveloper --db school
--collection student_tmp --type csv --headerline --file /students.csv
```

```
For mongosh info see: https://docs.mongodb.com/mongodb-shell/

test> use school
switched to db school
school> show collections
student
student_tmp
```

Contenido de student_tmp

db.student_tmp.find().pretty()

```
school> db.student_tmp.find().pretty()
[
  {
    _id: ObjectId("6313e715bdeddbc3ce0d3b6a"),
    id: 20920110,
    fullName: 'ANTONIO CARLOCK ANGEL FERNANDO',
    email: 'L20920110@voaxaca.tecnm.mx',
    career: 'INGENIERÍA INFORMÁTICA'
  },
  {
    _id: ObjectId("6313e715bdeddbc3ce0d3b6b"),
    id: 20920111,
    fullName: 'ARANGO LARA KEVIN JAVIER',
    email: 'L20920111@voaxaca.tecnm.mx',
    career: 'INGENIERÍA INFORMÁTICA'
  },
  {
    _id: ObjectId("6313e715bdeddbc3ce0d3b6c"),
    id: 'C17161059',
    fullName: 'AVENDAÑO SANCHEZ ABILENE GUADALUPE',
    email: 'LC17161059@voaxaca.tecnm.mx',
    career: 'INGENIERÍA INFORMÁTICA'
  },
  {
    _id: ObjectId("6313e715bdeddbc3ce0d3b6d"),
    id: 20920242,
    fullName: 'ARAGON ARAGON ELIANETH',
    email: 'L20920242@voaxaca.tecnm.mx',
    career: 'INGENIERÍA INFORMÁTICA'
  },
```

Exportar datos

```
mongoexport --authenticationDatabase admin --username
itvoDeveloper   --db school --collection student --out /students.json
```

Documentos incrustados en MongoDB

```
use school
db.employee.insertMany([
    {_id:1001,name:"Ambrosio",address:{previous:"Laureles
    S/N",current:"Laureles 121"},unit:"MongoDB"},
    {_id:1002,name:"Benedicto", address:{previous:"Emiliano Zapata
    199",current:"Ejutla de Crespo"},unit:"Java"},
    {_id:1003,name:"Kevin Molina",
    address:{previous:"Unknown",current:"Las Rosas 233 Col. Las
    flores"},unit:"Kotlin"}
])
```

Consultar documentos incrustados

db.employee.find({ address: { previous:"Laureles S/N",current:"Laureles 121" }})

```
school> db.employee.find( { address: { previous:"Laureles S/N",current:"Laureles 121" }} )
[
  {
    _id: 1001,
    name: 'Ambrosio',
    address: { previous: 'Laureles S/N', current: 'Laureles 121' },
    unit: 'MongoDB'
  }
]
school>
```

Consultar en campos anidados

db.employee.find({ "address.previous": "Unknown" })

```
school> db.employee.find( { "address.previous": "Unknown" } )
[
  {
    _id: 1003,
    name: 'Kevin Molina',
    address: { previous: 'Unknown', current: 'Las Rosas 233 Col. Las flores' },
    unit: 'Kotlin'
  }
]
school>
```

Manejo de arreglos

```
db.employeedetails.insertMany ([
        { name: "Ambrosio Cardoso", projects: ["MongoDB", "MariaDB","Oracle"], marks:[25,28,29] },
        { name: "Rodolfo Ibarra", projects: ["Python","TensorFlow"], marks:[26,24,23]},
        { name: "Maria del Carmen", projects: [ "PHP","MongoDB"], marks:[22,28,26]}
] )
```

```
db.employeedetails.find( { projects: ["MongoDB","MariaDB","Oracle"] } )
```

```
school> db.employeedetails.find( { projects: ["MongoDB","MariaDB","Oracle"] } )
[
  {
    _id: ObjectId("6313efc2c3eabcb070c35933"),
    name: 'Ambrosio Cardoso',
    projects: [ 'MongoDB', 'MariaDB', 'Oracle' ],
    marks: [ 25, 28, 29 ]
  }
]
school>
```

Listar los empleados que trabajan en proyecto MongoDB

db.employeedetails.find({ projects: "MongoDB" })

```
school> db.employeedetails.find( { projects: "MongoDB" } )
[
  {
    _id: ObjectId("6313efc2c3eabcb070c35933"),
    name: 'Ambrosio Cardoso',
    projects: [ 'MongoDB', 'MariaDB', 'Oracle' ],
    marks: [ 25, 28, 29 ]
  },
  {
    _id: ObjectId("6313efc2c3eabcb070c35935"),
    name: 'Maria del Carmen',
    projects: [ 'PHP', 'MongoDB' ],
    marks: [ 22, 28, 26 ]
  }
]
school>
```

Listar los empleados que tienen calificaciones mayores a 26

db.employeedetails.find({ marks:{$gt:26} })

Pruebe la siguiente instrucción:

db.employeedetails.find({ marks:{$gte:26} })

```
school> db.employeedetails.find( { marks:{$gt:26} } )
[
  {
    _id: ObjectId("6313efc2c3eabcb070c35933"),
    name: 'Ambrosio Cardoso',
    projects: [ 'MongoDB', 'MariaDB', 'Oracle' ],
    marks: [ 25, 28, 29 ]
  },
  {
    _id: ObjectId("6313efc2c3eabcb070c35935"),
    name: 'Maria del Carmen',
    projects: [ 'PHP', 'MongoDB' ],
    marks: [ 22, 28, 26 ]
  }
]
school>
```

Listar los empleados que tienen calificaciones mayores a 20 y menores que 24

db.employeedetails.find ({ marks: { $gt: 20, $lt: 24 } })

```
school> db.employeedetails.find( { marks: { $gt: 20, $lt: 24 } } )
[
  {
    _id: ObjectId("6313efc2c3eabcb070c35934"),
    name: 'Rodolfo Ibarra',
    projects: [ 'Python', 'TensorFlow' ],
    marks: [ 26, 24, 23 ]
  },
  {
    _id: ObjectId("6313efc2c3eabcb070c35935"),
    name: 'Maria del Carmen',
    projects: [ 'PHP', 'MongoDB' ],
    marks: [ 22, 28, 26 ]
  }
]
school>
```

Usando operador $elemMatch

$eleMatch permite especificar múltiples condiciones sobre los elementos del arreglo de tal manera que al menos un elemento de la matriz debe satisfacer todas las condiciones especificadas.

```
db.employeedetails.find(
{ marks: { $elemMatch: {
$gt: 23, $lt: 27 } } } )
```

```
school> db.employeedetails.find( { marks: { $elemMatch: { $gt: 23, $lt: 27 } } } )
[
  {
    _id: ObjectId("6313efc2c3eabcb070c35933"),
    name: 'Ambrosio Cardoso',
    projects: [ 'MongoDB', 'MariaDB', 'Oracle' ],
    marks: [ 25, 28, 29 ]
  },
  {
    _id: ObjectId("6313efc2c3eabcb070c35934"),
    name: 'Rodolfo Ibarra',
    projects: [ 'Python', 'TensorFlow' ],
    marks: [ 26, 24, 23 ]
  },
  {
    _id: ObjectId("6313efc2c3eabcb070c35935"),
    name: 'Maria del Carmen',
    projects: [ 'PHP', 'MongoDB' ],
    marks: [ 22, 28, 26 ]
  }
]
school>
```

Buscar por la posición del índice

db.employeedetails.find({ "marks.2": { $gt: 26 } })

```
school> db.employeedetails.find( { "marks.2": { $gt: 26 } } )
[
  {
    _id: ObjectId("6313efc2c3eabcb070c35933"),
    name: 'Ambrosio Cardoso',
    projects: [ 'MongoDB', 'MariaDB', 'Oracle' ],
    marks: [ 25, 28, 29 ]
  }
]
school>
```

Usando operador $size

db.employeedetails.find({ "projects": { $size: 3 } })

```
school> db.employeedetails.find( { "projects": { $size: 3 } } )
[
  {
    _id: ObjectId("6313efc2c3eabcb070c35933"),
    name: 'Ambrosio Cardoso',
    projects: [ 'MongoDB', 'MariaDB', 'Oracle' ],
    marks: [ 25, 28, 29 ]
  }
]
```

Usando operador $push

db.employeedetails.updateOne({name:"Ambrosio Cardoso"},{$push:{projects: "Postgres"}})

```
school> db.employeedetails.find( { "projects": { $size: 4 } } )
[
  {
    _id: ObjectId("6313efc2c3eabcb070c35933"),
    name: 'Ambrosio Cardoso',
    projects: [ 'MongoDB', 'MariaDB', 'Oracle', 'Postgres' ],
    marks: [ 25, 28, 29 ]
  }
]
school>
```

Usando operador $each

db.employeedetails.update({name:"Ambrosio Cardoso"}, {$push:{languages: {$each:["Mixe","Español","Ingles"]}}})

```
school> db.employeedetails.find({name:"Ambrosio Cardoso"})
[
  {
    _id: ObjectId("6313efc2c3eabcb070c35933"),
    name: 'Ambrosio Cardoso',
    projects: [ 'MongoDB', 'MariaDB', 'Oracle', 'Postgres' ],
    marks: [ 25, 28, 29 ],
    languages: [ 'Mixe', 'Español', 'Ingles' ]
  }
]
school>
```

Usando operador $pop

```
db.employeedetails.update({name:"Ambrosio Cardoso"},{$pop:{projects:1}})
```

Usando operador $addToSet

Agrega elementos a un arreglo sin duplicidad

```
db.employeedetails.updateOne( {name: "Ambrosio Cardoso"}, { $addToSet:{hobbies: ["Programming", "sing", "Play Basketball","Play chess","Write Poem"]} })
```

Consultar datos en arreglos anidados

```
db.studentmarks.insertMany([
{ name:"Rosa",
  marks:[
          {class: "II", total: 489},
          { class: "III", total: 490 }
  ]
},
{name:"Marco",
        marks:[
                {class: "III", total: 469 },
                {class: "IV",  total: 450}
        ]
},
{name:"Jesus",
        marks:[
                {class:"II", total: 489 },
                {class: "III", total: 390}
        ]
},
{name:"Flor",
        marks:[
                {class:"III", total: 489},
                {class: "IV", total: 490}
        ]
},
{name:"Axel",
        marks:[
                {class: "II", total: 465},
                { class: "III",total: 470}
        ]
}
])
```

Consultar datos en arreglos anidados

db.studentmarks.find({ "marks": {class: "II", total: 489}})

```
school> db.studentmarks.find( { "marks": {class: "II", total: 489}})
[
  {
    _id: ObjectId("63180a0de0e6de5b7cde6d65"),
    name: 'Rosa',
    marks: [ { class: 'II', total: 489 }, { class: 'III', total: 490 } ]
  },
  {
    _id: ObjectId("63180a0de0e6de5b7cde6d67"),
    name: 'Jesus',
    marks: [ { class: 'II', total: 489 }, { class: 'III', total: 390 } ]
  }
]
school>
```

Consultar datos en arreglos anidados

db.studentmarks.find({ 'marks.total': { $lt: 400 } })

```
school> db.studentmarks.find( { 'marks.total': { $lt: 400 } } )
[
  {
    _id: ObjectId("63180a0de0e6de5b7cde6d67"),
    name: 'Jesus',
    marks: [ { class: 'II', total: 489 }, { class: 'III', total: 390 } ]
  }
]
school>
```

Consultar datos en arreglos anidados

**db.studentmarks.
find({
'marks.0.class': "II" }
)**

```
school> db.studentmarks.find( { 'marks.0.class': "II" } )
[
  {
    _id: ObjectId("63180a0de0e6de5b7cde6d65"),
    name: 'Rosa',
    marks: [ { class: 'II', total: 489 }, { class: 'III', total: 490 } ]
  },
  {
    _id: ObjectId("63180a0de0e6de5b7cde6d67"),
    name: 'Jesus',
    marks: [ { class: 'II', total: 489 }, { class: 'III', total: 390 } ]
  },
  {
    _id: ObjectId("63180a0de0e6de5b7cde6d69"),
    name: 'Axel',
    marks: [ { class: 'II', total: 465 }, { class: 'III', total: 470 } ]
  }
]
school>
```

Genere una nueva colección

```
db.studentdetails.insertMany( [
{       name: "Rosa", result: "pass",
        marks: { english: 25, maths: 23, science: 25 },
        grade: [ {class: "A", total: 73 } ]
},
{       name: "Marco", result: "pass",
        marks: { english: 24, maths: 25, science: 25 },
        grade: [ {class: "A", total: 74 } ]
},
{       name: "Marco", result: "fail",
        marks: { english: 12, maths: 13, science: 15 },
        grade: [ {class: "C", total: 40 } ]
}
])
```

Restringiendo campos que devuelve una consulta

db.studentdetails.find({ name: "Rosa" }, { marks: 1, result: 1 })

```
school> db.studentdetails.find( { name: "Rosa" }, { marks: 1, result: 1 })
[
  {
    _id: ObjectId("63180d01e0e6de5b7cde6d6a"),
    result: 'pass',
    marks: { english: 25, maths: 23, science: 25 }
  }
]
school>
```

db.studentdetails.find({ name: "Rosa" }, { marks: 1, result: 1,_id:0 })

```
school> db.studentdetails.find( { name: "Rosa" }, { marks: 1, result: 1,_id:0 } )
[ { result: 'pass', marks: { english: 25, maths: 23, science: 25 } } ]
school>
```

Restringiendo campos que devuelve una consulta

db.studentdetails.find({ name: "Rosa" }, { result:1, grade: 1, "marks.english": 1 })

```
school> db.studentdetails.find( { name: "Rosa" }, { result:1, grade: 1, "marks.english": 1 })
[
  {
    _id: ObjectId("63180d01e0e6de5b7cde6d6a"),
    result: 'pass',
    marks: { english: 25 },
    grade: [ { class: 'A', total: 73 } ]
  }
]
school>
```

Filtra aquellos documentos que tienen el campo fullName

```
db.studentdetails.find( { fullName: { $exists: true } } )
```

Uso de cursores

```
db.numbers.insertOne({_id:1,number:1});
db.numbers.insertOne({_id:2,number:2});
db.numbers.insertOne({_id:3,number:3});
db.numbers.insertOne({_id:4,number:4});
db.numbers.insertOne({_id:5,number:5});
db.numbers.insertOne({_id:6,number:6});
db.numbers.insertOne({_id:7,number:7});
db.numbers.insertOne({_id:8,number:8});
db.numbers.insertOne({_id:9,number:9});
db.numbers.insertOne({_id:10,number:10});
db.numbers.insertOne({_id:11,number:11});
db.numbers.insertOne({_id:12,number:12});
db.numbers.insertOne({_id:13,number:13});
db.numbers.insertOne({_id:14,number:14});
db.numbers.insertOne({_id:15,number:15});
```

```
db.numbers.insertOne({_id:16,number:16});
db.numbers.insertOne({_id:17,number:17});
db.numbers.insertOne({_id:18,number:18});
db.numbers.insertOne({_id:19,number:19});
db.numbers.insertOne({_id:20,number:20});
db.numbers.insertOne({_id:21,number:21});
db.numbers.insertOne({_id:22,number:22});
db.numbers.insertOne({_id:23,number:23});
db.numbers.insertOne({_id:24,number:24});
db.numbers.insertOne({_id:25,number:25});
```

Uso de cursores

```
var myCursor=db.numbers.find({});

while (myCursor.hasNext()) {
        print(myCursor.next());
}
```

```
school> var myCursor=db.numbers.find({});

school> while (myCursor.hasNext()) { print(myCursor.next()); }
{ _id: 1, number: 1 }
{ _id: 2, number: 2 }
{ _id: 3, number: 3 }
{ _id: 4, number: 4 }
{ _id: 5, number: 5 }
{ _id: 6, number: 6 }
{ _id: 7, number: 7 }
{ _id: 8, number: 8 }
{ _id: 9, number: 9 }
{ _id: 10, number: 10 }
{ _id: 11, number: 11 }
{ _id: 12, number: 12 }
{ _id: 13, number: 13 }
{ _id: 14, number: 14 }
{ _id: 15, number: 15 }
{ _id: 16, number: 16 }
{ _id: 17, number: 17 }
{ _id: 18, number: 18 }
{ _id: 19, number: 19 }
{ _id: 20, number: 20 }
{ _id: 21, number: 21 }
{ _id: 22, number: 22 }
{ _id: 23, number: 23 }
{ _id: 24, number: 24 }
{ _id: 25, number: 25 }

school>
```

Uso de cursores

```
var myCursor=db.numbers.find({});

myCursor.forEach(printjson);
```

```
school> var myCursor=db.numbers.find({});

school> myCursor.forEach(printjson);
{ _id: 1, number: 1 }
{ _id: 2, number: 2 }
{ _id: 3, number: 3 }
{ _id: 4, number: 4 }
{ _id: 5, number: 5 }
{ _id: 6, number: 6 }
{ _id: 7, number: 7 }
{ _id: 8, number: 8 }
{ _id: 9, number: 9 }
{ _id: 10, number: 10 }
{ _id: 11, number: 11 }
{ _id: 12, number: 12 }
{ _id: 13, number: 13 }
{ _id: 14, number: 14 }
{ _id: 15, number: 15 }
{ _id: 16, number: 16 }
{ _id: 17, number: 17 }
{ _id: 18, number: 18 }
{ _id: 19, number: 19 }
{ _id: 20, number: 20 }
{ _id: 21, number: 21 }
{ _id: 22, number: 22 }
{ _id: 23, number: 23 }
{ _id: 24, number: 24 }
{ _id: 25, number: 25 }

school>
```

Trabajando con método limit() y skip()

db.numbers.find().limit(2) → **[{ _id: 1, number: 1 }, { _id: 2, number: 2 }]**

db.numbers.find().skip(2) →

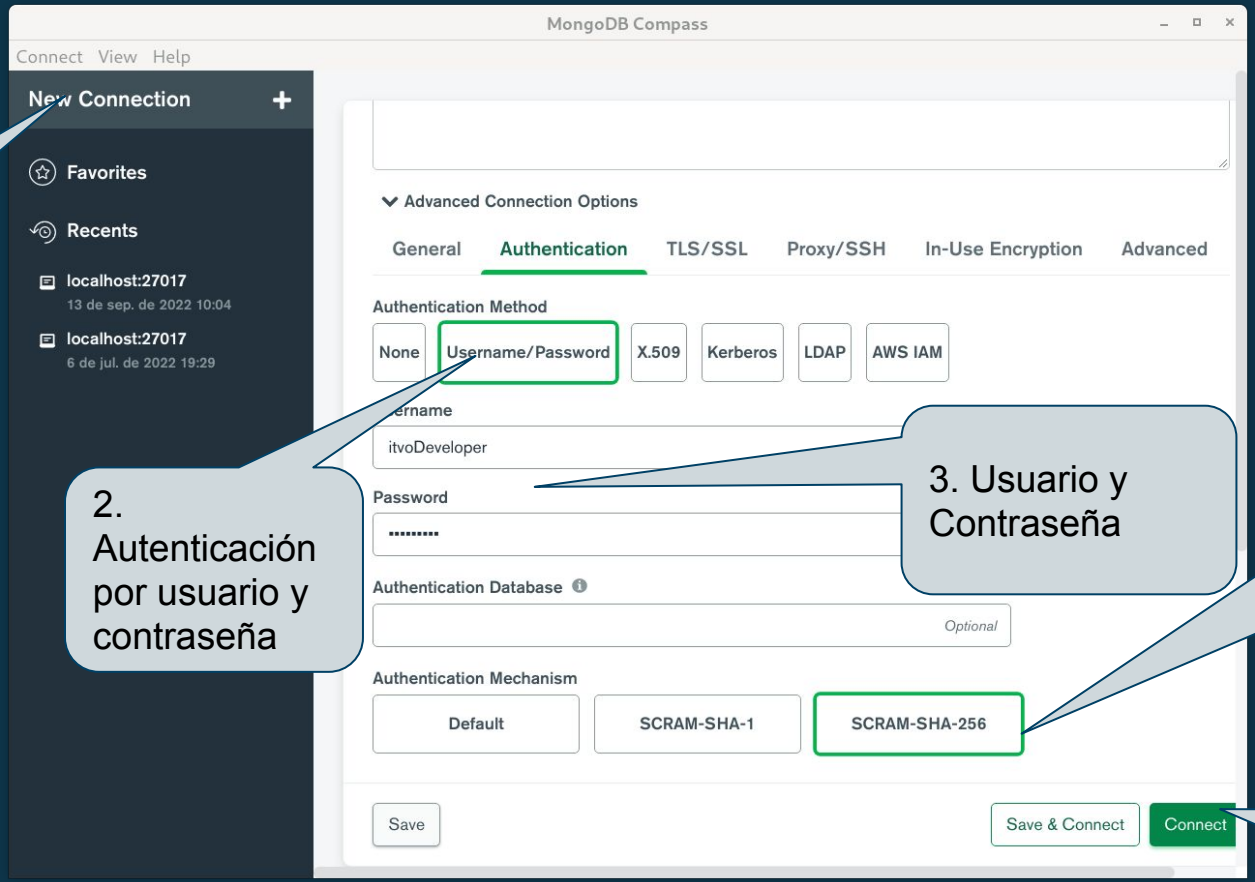
```
school> db.numbers.find().skip(2)
[
  { _id: 3, number: 3 },   { _id: 4, number: 4 },
  { _id: 5, number: 5 },   { _id: 6, number: 6 },
  { _id: 7, number: 7 },   { _id: 8, number: 8 },
  { _id: 9, number: 9 },   { _id: 10, number: 10 },
  { _id: 11, number: 11 }, { _id: 12, number: 12 },
  { _id: 13, number: 13 }, { _id: 14, number: 14 },
  { _id: 15, number: 15 }, { _id: 16, number: 16 },
  { _id: 17, number: 17 }, { _id: 18, number: 18 },
  { _id: 19, number: 19 }, { _id: 20, number: 20 },
  { _id: 21, number: 21 }, { _id: 22, number: 22 }
]
Type "it" for more
school> it
[
  { _id: 23, number: 23 },
  { _id: 24, number: 24 },
  { _id: 25, number: 25 }
]
school>
```

Mongo compass

Compass es una herramienta interactiva para consultar, optimizar y analizar datos en MongoDB

https://www.mongodb.com/try/download/compass

Configurar conexión



1. Nueva conexión

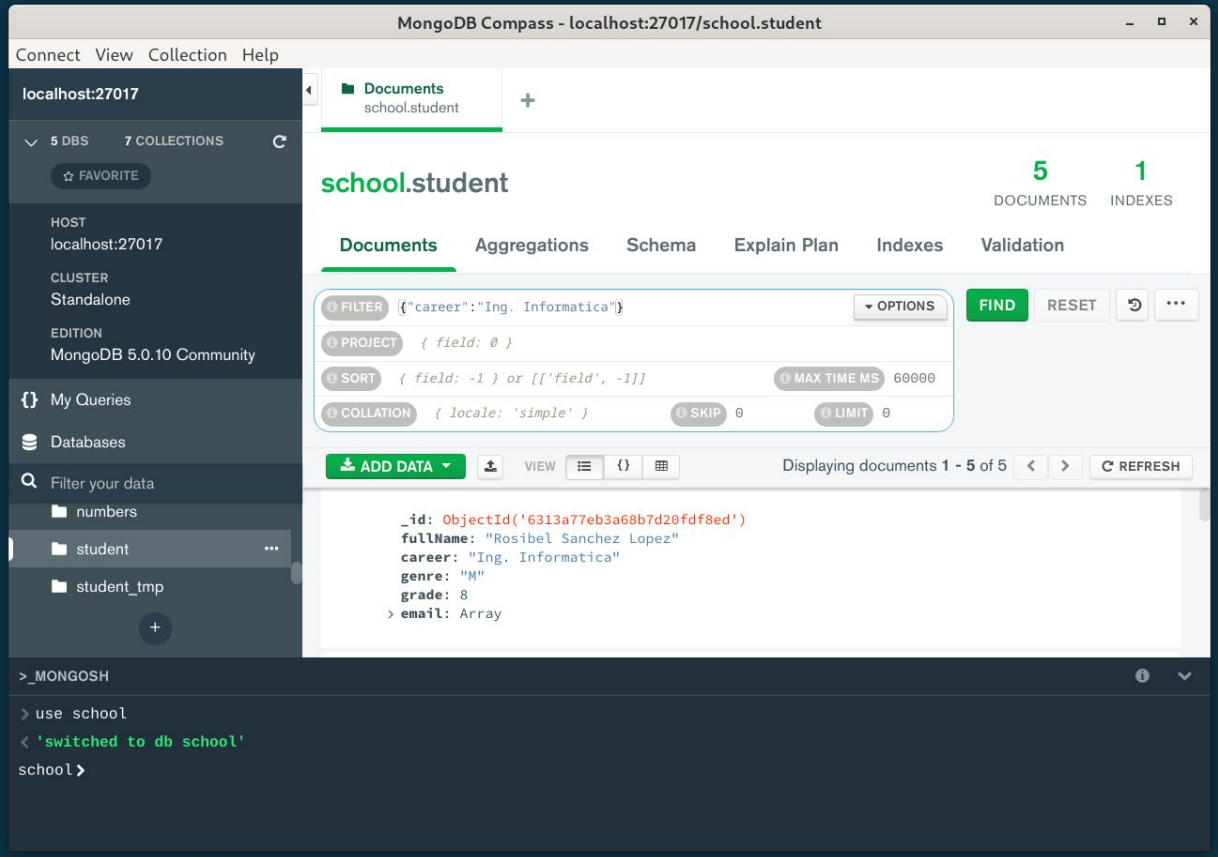
2. Autenticación por usuario y contraseña

3. Usuario y Contraseña

4. Mecanismo de autenticación

5. Conectar

Acciones en MongoDB



Aggregation

- **Aggregation pipeline.**
- **Map-reduce function (En desuso)**
- **Single-purpose aggregation methods.**

Insertar datos para usarlo en las agregaciones

```
db.orders.insertMany([
{custID:10001,amount:500,status:"A"},
{custID:10001,amount:250,status:"A"},
{custID:10002,amount:200,status:"A"},
{custID:10001,amount: 300, status:"D"}
]);
```

Aggregation pipeline

db.orders.aggregate({$group:{_id:"$custID",TotalAmount:{$sum:"$amount"}}});

```
>_MONGOSH
< { custID: 10001 }
  { custID: 10001 }
  { custID: 10002 }
  { custID: 10001 }
> db.orders.aggregate({$group:{_id:"$custID",TotalAmount:{$sum:"$amount"}}});
< { _id: 10001, TotalAmount: 1050 }
  { _id: 10002, TotalAmount: 200 }
school>
```

Aggregation pipeline

db.orders.aggregate({$match:{status:"A"}},{$group:{_id:"$custID",TotalAmount:{$sum:"$amount"}}});

```
>_MONGOSH

> db.orders.aggregate({$match:{status:"A"}},{$group:{_id:"$custID",TotalAmount:{$sum:"$amount"}}})
< { _id: 10001, TotalAmount: 750 }
  { _id: 10002, TotalAmount: 200 }
school>
```

Aggregation pipeline (AVG)

db.orders.aggregate({$group:{_id:"$custID",
AverageAmount:{$avg:"$amount"}}});

```
>_MONGOSH

> db.orders.aggregate({$group:{_id:"$custID",AverageAmount:{$avg:"$amount"}}});
< { _id: 10002, AverageAmount: 200 }
  { _id: 10001, AverageAmount: 350 }
school>
```

Single-Purpose Aggregation Operations

```
db.orders.distinct("custID")
db.orders.estimatedDocumentCount()
```

SQL	MongoDB Operator
WHERE	$match
GROUP BY	$group
HAVING	$match
SELECT	$project
ORDER BY	$sort
LIMIT	$limit
SUM	$sum
COUNT	$sum
JOIN	$lookup

Conversión de operaciones de agregación SQL a MongoDB

```
SELECT *
FROM orders;
```

```
db.orders.find()
```

```
SELECT custID, amount
FROM orders
```

```
db.orders.find({}, {_id:0, custID:1,
amount:1})
```

Conversión de operaciones de agregación SQL a MongoDB

```
SELECT *
FROM orders
WHERE status='A'
```

```
db.orders.find({status:'A'})
```

```
SELECT *
FROM orders
WHERE status<>'A'
```

```
db.orders.find({status:{$ne:'A'}})
```

Conversión de operaciones de agregación SQL a MongoDB

```
SELECT *
FROM orders
WHERE status='A'
        OR amount <300
```

```
db.orders.find({$or:[
{status:'A'}, {amount:{$lt:300}}]})
```

```
SELECT *
FROM orders
WHERE status='A'
        AND amount <300
```

```
db.orders.find({status:'A',
amount:{$lt:300}})
```

Conversión de operaciones de agregación SQL a MongoDB

```
SELECT *
FROM orders
WHERE status='A'
        OR amount <300
ORDER BY amount DESC
```

```
db.orders.find({$or:[
{status:'A'},
{amount:{$lt:300}}]}).sort({amount:-1}
)
```

DESC

Conversión de operaciones de agregación SQL a MongoDB

```
SELECT COUNT(*) AS count
FROM orders;
```

```
db.orders.aggregate( [
{ $group: {  _id: null,
count: {$sum: 1 } } }
] )
```

```
SELECT custID, sum(amount) AS
TotalAmount
FROM orders
GROUP BY custID
WHERE status='A';
```

```
db.orders.aggregate(
{$match:{status:"A"}},
{$group:{_id:"$custID",TotalAmount:
{$sum:"$amount"}}});
```

Conversión de operaciones de agregación SQL a MongoDB

```
SELECT custID, sum(amount) AS
TotalAmount
FROM orders
GROUP BY custID
WHERE status='A'
ORDER BY TotalAmount DESC
```

```
db.orders.aggregate(
{$match:{status:"A"}},
{$group:{_id:"$custID",TotalAmount:
{$sum:"$amount"}}},
{$sort: {"TotalAmount":-1} }
)
```

¡Gracias!

Ambrosio Cardoso Jiménez

ambrosio.cj@voaxaca.tecnm.mx

Septiembre 2022