

**CÁTEDRA ADMINISTRACIÓN DE BASES DE DATOS**  
UNIVERSIDAD TECNOLÓGICA NACIONAL – FACULTAD REGIONAL LA PLATA  
- INGENIERÍA EN SISTEMAS DE INFORMACIÓN –

# Bases de Datos Documentales

**NoSQL - MongoDB**

# ¿Qué es un MongoDB?



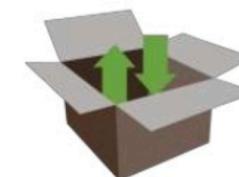
**MongoDB** (del inglés humongous, "enorme") es un sistema de base de datos NoSQL orientado a documentos de código abierto.



Fully Featured  
High Performance  
Scalable

```
{  
  name: "John Smith",  
  pfxs: ["Dr.", "Mr."],  
  address: "10 3rd St.",  
  phones: [  
    { number: "555-1212",  
      type: "land" },  
    { number: "444-1212",  
      type: "mobile" }  
  ]  
}
```

Document  
Data Model



Open-  
Source

# ¿Qué es un MongoDB? (cont.)



500+ employees



2000+ customers



**Offices** in NY & Palo Alto and across EMEA, and APAC



Over \$311 million in funding

## Características principales

- No necesita *schemas* para guardar los datos, ya que los guarda de forma flexible en **documentos** con estructuras similares a un JSON.
- La **estructura de cada documento puede variar**, y la estructura de los datos también. Flexible.
- Alto nivel.

## Características principales

- Estos documentos **se mapean a objetos** para ser utilizados a nivel aplicación, lo que hace que se pueda trabajar con los datos de manera amigable, permitiendo *queries* y agregaciones en tiempo real de los datos.
- Está pensada para ser **distribuida**, permitiendo alta disponibilidad y escalabilidad horizontal.

# Los datos son el schema

Relational

Customer ID	First Name	Last Name	City
0	John	Doe	New York
1	Mark	Smith	San Francisco
2	Jay	White	Dallas
3	Meagan	White	London
4	Edward	Daniels	Boston

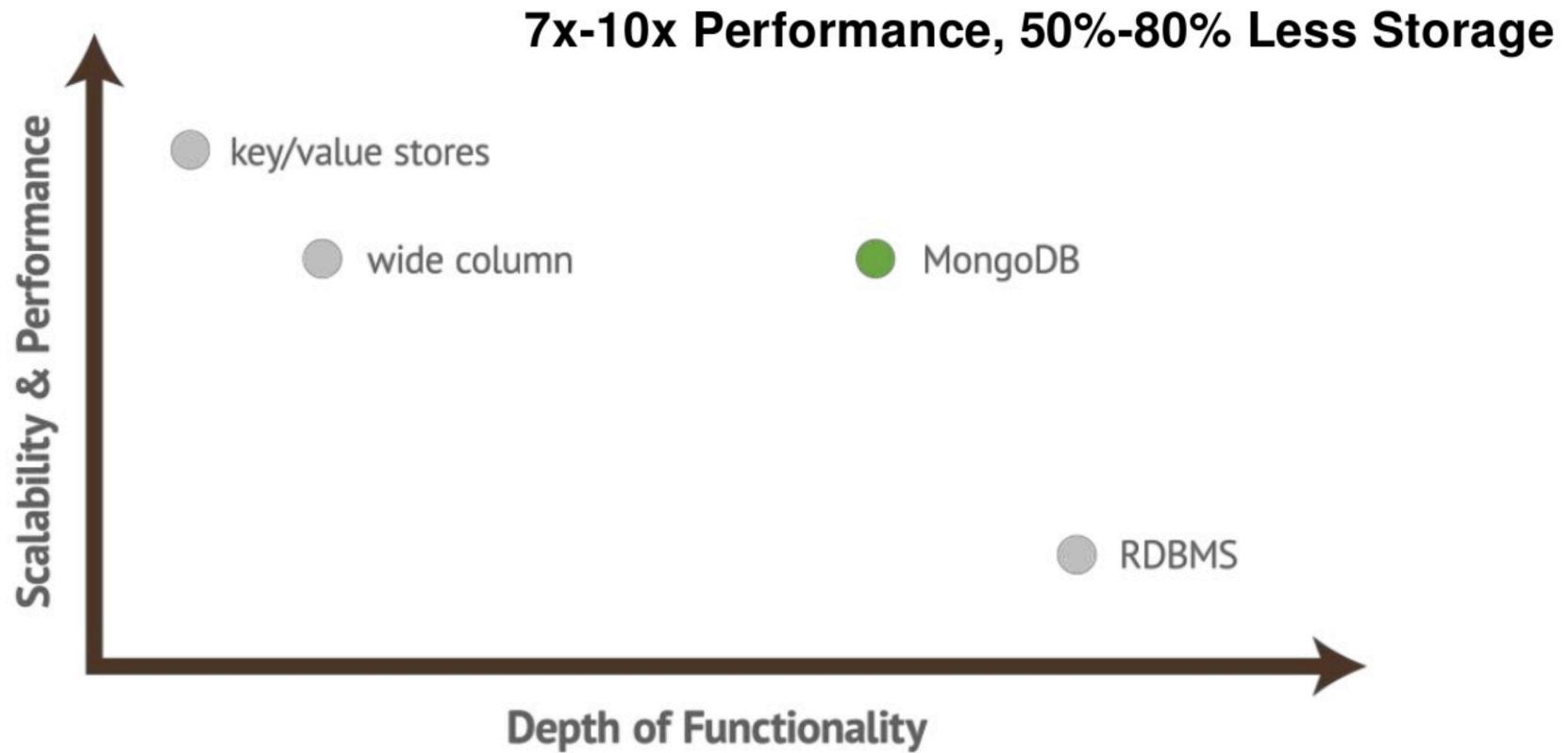
Phone Number	Type	DNC	Customer ID
1-212-555-1212	home	T	0
1-212-555-1213	home	T	0
1-212-555-1214	cell	F	0
1-212-777-1212	home	T	1
1-212-777-1213	cell	(null)	1
1-212-888-1212	home	F	2

MongoDB

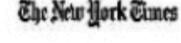


```
{ "vers": 1,  
  customer_id : 1,  
  name : {  
    "f": "Mark",  
    "l": "Smith" },  
  city : "San Francisco",  
  phones: [ {  
    number : "1-212-777-1212",  
    dnc : true,  
    type : "home"  
  },  
  {  
    number : "1-212-777-1213",  
    type : "cell"  
  } ]}
```

## Características principales (cont.)



# Casos de uso

Big Data	Product & Asset Catalogs	Security & Fraud	Internet of Things	Database-as-a-Service	Complex Data Management
  	   	  Top Investment and Retail Banks Intelligence Agencies	Top Global Shipping Company  	 	Cushman & Wakefield Top Investment and Retail Banks
Mobile Apps	Customer Data Management	Single View	Social & Collaboration	Content Management	Embedded / ISV
  	    	  	   	   	 

# ¿Quién lo usa?

facebook

invision

ebay™

Adobe

Google

SQUARESPACE

coinbase

SEGA®

intuit®

eharmony

EA™

verizon^

shutterfly

GOV.UK

SAP®

# Terminología

RDBMS	MongoDB
Database	Database
Table	Collection
Index	Index
Row	Document
Column	Field
Join	Embedding & Linking & \$lookup

# Drivers

Factory



Community



# Instalación

- <https://docs.mongodb.com/manual/administration/install-community/>

The screenshot shows a browser window for the MongoDB Documentation website. The URL in the address bar is <https://docs.mongodb.com/manual/administration/install-community/>. The page title is "Install MongoDB Community Edition". The top navigation bar includes links for SERVER, DRIVERS, CLOUD, TOOLS, GUIDES, and Get MongoDB, along with a search bar. The left sidebar, titled "MONGDB MANUAL" and "4.2 (current)", contains sections for Introduction, Installation (with "Install MongoDB Community Edition" expanded), and other topics like The mongo Shell and MongoDB CRUD Operations. The main content area describes the installation process for MongoDB Community Edition, with sections for Linux, macOS, and Windows, each providing a link to the specific installation guide.

- <https://docs.mongodb.com/manual/tutorial/manage-mongodb-processes/>

# Consola MongoDB (*mongo shell*)

Debe estar el ***mongod*** en ejecución.

```
$ mongod --port 27017 --dbpath /data
```

```
$ mongo
```

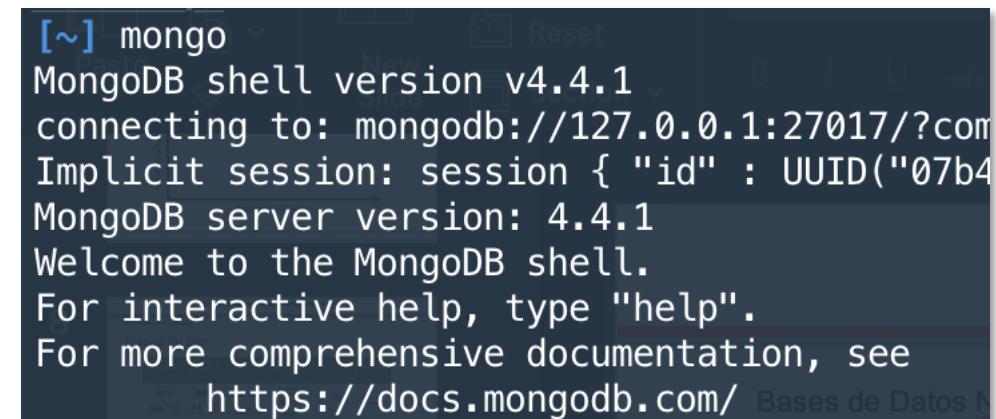
```
$ mongo --versión
```

```
$ mongo --host 10.10.8.1 --port 27017
```

```
    > help
```

```
    > quit()
```

***mongod*** es el proceso servidor;  
***mongo*** es el cliente



```
[~] mongo
MongoDB shell version v4.4.1
connecting to: mongodb://127.0.0.1:27017/?com
Implicit session: session { "id" : UUID("07b4
MongoDB server version: 4.4.1
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
https://docs.mongodb.com/ Bases de Datos N
```

## Creación de base de datos

MongoDB no posee un comando para crear base de datos.  
Se debe “seleccionar”.

```
> use Ejemplos  
> show dbs  
> db
```

```
# apagar el servidor mongod  
> db.shutdownServer()
```

## Eliminar base de datos

El comando *dropDatabase()* elimina la base de datos seleccionada y todos sus datos asociados.

```
> db.dropDatabase()
```

## Creación de colecciones

MongoDB guarda a los documentos en colecciones, que podrían entenderse análogamente como tablas.

Aunque no es necesaria la creación (se crean al agregar un documento) existe un comando para ello.

```
> db.createCollection('Peliculas')
> show collections
> db.Peliculas.drop()
```

## Insertando documentos

El método *insert()* sirve para agregar documentos a una colección. Dicha colección puede no estar previamente creada.

Se pueden agregar documentos individuales o en grupo.

```
> db.Peliculas.insert({})
> db.Peliculas.insert({'test': 123})
> db.Peliculas.insert({'otro': false})
> db.Peliculas.insertMany([{...}, {...}, {...}])
```

## Insertando documentos (cont.)

```
$ mongoimport --type json --db ejemplos --collection Alumnos --file students.json
```

```
$ mongoimport --type csv --headerline \  
  --db ejemplos \  
  --collection Peliculas \  
  --file peliculas.csv
```

<https://docs.mongodb.com/manual/reference/program/mongoimport/>

Para generar backups se utilizará **mongorestore**:

<https://docs.mongodb.com/manual/reference/program/mongorestore/>

# Primary Key en MongoDB

En MongoDB, para identificar únicamente a cada documento dentro de la colección se agrega un campo **\_id** automáticamente al crearlo, al menos que el usuario lo cargue.

El tipo de dicho campo será **ObjectId**.

Sí, MongoDB tiene índices.

```
{  
    "_id" : ObjectId("5db7c6ecc79cc1621f13278e"),  
    "codigo_imdb" : "tt1535109",  
    "nombre" : "Captain Phillips",  
    "presupuesto" : 55000000,  
    "recaudacion" : 218743570  
}  
{  
    "_id" : ObjectId("5db7c6ecc79cc1621f13278f"),  
    "codigo_imdb" : "tt1985966",  
    "nombre" : "Cloudy with a Chance of Meatballs 2",  
    "presupuesto" : 78000000,  
    "recaudacion" : 271725448  
}  
{  
    "_id" : ObjectId("5db7c6ecc79cc1621f132790"),  
    "codigo_imdb" : "tt1690953",  
    "nombre" : "Despicable Me 2",  
    "presupuesto" : 76000000,  
    "recaudacion" : 970766005  
}
```

## Consulta de documentos

Para consultar documentos se utiliza el método *find()*:

```
> db.Peliculas.find()  
> db.Peliculas.find().pretty()  
> db.Peliculas.find({<condiciones>}, {<campos, proyección>} ).pretty()
```

SELECT \* FROM Peliculas

SELECT <campos, proyección> FROM Peliculas WHERE <condiciones>

<https://docs.mongodb.com/manual/reference/operator/query/>

## Edición de documentos

Para editar documentos se utiliza el método *update()*:

```
> db.Peliculas.update(  
    {<condiciones>},  
    {<nuevos datos>}  
)
```

## Eliminación de documentos

Para eliminar documentos se utiliza el método *remove()*. Si no se utilizan condiciones, se eliminarán todos los documentos de la colección.

```
> db.Peliculas.remove({<condiciones>})
> db.Peliculas.remove({ name: "Gisela Levin" })
> db.Peliculas.remove({ _id: "..." })
> db.Peliculas.remove({ _id: ObjectId("...") })
>db.Peliculas.remove()
```

## Complementos para consulta de documentos

Para eliminar documentos se utiliza el método *remove()*. Si no se utilizan condiciones, se eliminarán todos los documentos de la colección.

```
> db.Peliculas.find().limit(<cantidad>)
> db.Peliculas.find().limit(<cantidad>).skip(<cantidad>)
> db.Peliculas.find().sort(<campo>: 1) # ó -1
> db.Peliculas.count()
> db.Peliculas.find().count()
```

# Usuarios y Control de acceso

MongoDB en la instalación base no contiene usuarios ni control de acceso.

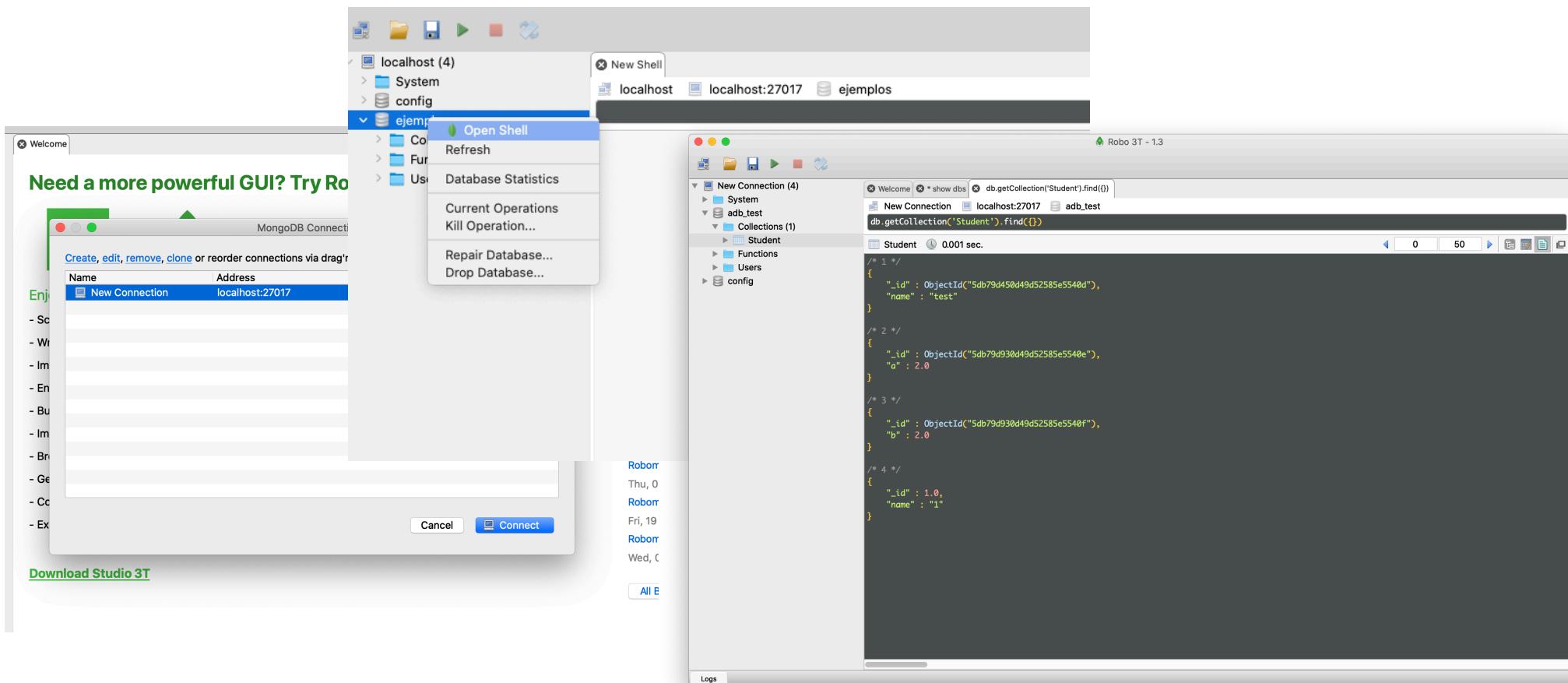
```
> use admin
> db.createUser({
  user: "myUserAdmin",
  pwd: "abc123",
  roles: [{ role: "userAdminAnyDatabase", db: "admin" }]
})
```

```
> mongod --auth --port 27017 --dbpath /data/db1
> mongo -u "myUserAdmin" -p "abc123" \
--authenticationDatabase "admin"
```

<https://docs.mongodb.com/manual/reference/method/js-user-management/>

# Interfaces gráficas

- Robo 3T (ex-Robomongo) <https://robomongo.org/>



# Interfaces gráficas

- MongoDB Compass

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

The screenshot shows the MongoDB Compass interface with the following details:

- Left Sidebar:** Shows "My Cluster" with 15 DBs and 34 Collections. Databases listed include admin, app, auth, config, employees, foo, local, log, mdbw, metadata, nylphil, and schemaDb.
- Central View:** Shows "Compass Test Cluster" with 4 DBs and 9 Collections. It displays a list of databases with their storage sizes, collections, and indexes.
- Right Panel:** Shows the "travel.airports" collection with 12.3k documents. It includes tabs for Documents, Aggregations, Schema, Explain Plan, Indexes, and Validation. A search bar filters results for "Country: 'Brazil'".
- Document Preview:** Two documents are shown in detail:
  - Airport 1:** Located in Conceição do Araguaia, Brazil. Details include IATA: CDJ, ICAO: SBAA, Latitude: -8.348349571228027, Longitude: -49.30149841308594, Altitude: 653m, Timezone: -3, DST: -5, Tz database time zone: America/Belem, Type: airport, Source: OurAirports.
  - Airport 2:** Located in Campo Délío Jardim de Mattos, Rio De Janeiro, Brazil. Details include IATA: N, ICAO: SBAF, Latitude: -22.875099, Longitude: -43.384701, Altitude: 118m, Timezone: -3, DST: -5, Tz database time zone: America/Rio\_Beira\_Mar, Type: airport, Source: OurAirports.

# Operadores

Se pueden utilizar distintos operadores para lograr consultas más complejas.

<https://docs.mongodb.com/manual/reference/operator/query/#query-selectors>

```
{ <campo>: { <operator>: <value> }, ... }
```

```
> db.Peliculas.find({ nombre: "Frozen" }).pretty()
> db.Peliculas.find({ nombre: {
    $in: [ "Frozen", "The Host" ]
}})
```

```
SELECT * FROM <tabla>
WHERE nombre in ("Frozen", "The Host")
```

# LIKE

# Trae películas que comiencen con “star”

```
> db.Peliculas.find({ nombre: /^star /i }).pretty()
```

```
> db.Peliculas.find({ nombre: { $regex: /^star /i } } )
```

SELECT \* FROM Peliculas

WHERE nombre LIKE "Star Wars%"

# Trae películas que terminen en “ars”

```
> db.Peliculas.find({ nombre: /ars$/ }).pretty()
```

SELECT \* FROM Peliculas

WHERE nombre LIKE "%ars"

<https://docs.mongodb.com/manual/reference/operator/query/regex/>

## AND

# Trae películas de nombre "Star Wars" y presupuesto mayor a 1.000.000.

```
> db.Peliculas.find({  
    nombre: "Star Wars",  
    presupuesto: {  
        $gt: 1000000  
    }  
})
```

```
SELECT * FROM Peliculas  
WHERE nombre = "Star Wars" AND presupuesto > 30
```

## OR

# Trae todas las películas de nombre "Star Wars" o presupuesto mayor a 1.000.000 y menor a 5.000.000.

```
> db.Peliculas.find({ $or: [
    { nombre: "Star Wars"},  

    { presupuesto: {  

        $gt: 1000000,  

        $lt: 5000000
    }}
]})
```

```
SELECT * FROM Peliculas  
WHERE nombre = "Star Wars" OR presupuesto > 1000000
```

## AND y OR

# Trae películas de nombre "Star Wars" y presupuesto mayor a 1.000.000 ó recaudación mayor a 500.000.

```
> db.Peliculas.find({  
    nombre: "Star Wars",  
    $or: [  
        { presupuesto: { $gt: 1000000 } },  
        { recaudacion: { $gt: 500000 } }  
    ]  
})
```

```
SELECT * FROM Peliculas  
WHERE nombre = "Star Wars"  
AND ( presupuesto > 1000000 OR recaudación > 500000 )
```

# DISTINCT

# Trae todos los “nombres” únicos

```
> db.Peliculas.distinct("nombre")  
> db.Alumnos.distinct("name")
```

# Trae los “type” únicos dentro de “scores”

```
> db.Alumnos.distinct("scores.type")
```

# Trae los “type” únicos dentro de “scores”, de “\_id” entre 5 y 10

```
> db.Alumnos.distinct("scores.type", { _id: { $gt: 5, $lt: 10 }})
```

## Existencia

# Trae todos los estudiantes sin “lastname”

```
> db.Alumnos.find({ lastname: { $exists: 0 } })
```

```
> db.Alumnos.find({ lastname: { $exists: false } })
```

# Idem, pero sólo mostrando “name”

```
> db.Alumnos.find({ lastname: { $exists: 0 } }, { name: 1 })
```

## Condiciones en campos anidados

# Trae todos los estudiantes con por lo menos 90 en un examen  
> db.Alumnos.find({ "scores.score": { \$gt: 90 } })

# ¿Qué pasa acá?  
> db.Alumnos.find({  
    "scores.type": "quiz",  
    "scores.score": { \$gt: 90 }  
})

→ MongoDB verifica cada condición para cada valor de la lista. Si todas las condiciones se satisfacen al menos por un valor, traerá ese documento.

## \$elemMatch

```
> db.Alumnos.find({  
    "scores.type": "quiz",  
    "scores.score": { $gt: 90 }  
})
```

# Con *\$elemMatch*, se buscarán documentos que cumplan todas las condiciones en por lo menos un valor de la lista.

```
> db.Alumnos.find({  
    scores: { $elemMatch: {  
        type: "quiz",  
        score: { $gt: 90 }  
    }}  
})
```

## Agregación – *aggregate()*

Las agregaciones son operaciones que nos ayudan a procesar datos y devolver resultados computados.

```
> db.<colección>.aggregate([<operaciones de agregación>])
```

# Operaciones de agregación

<https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/>

- **\$match** – Filtra documentos que cumplan con el criterio.
- **\$project** – Selecciona/crea campos en los documentos.
- **\$unwind** – Separa elementos de una lista de valores en documentos separados.
- **\$sort** – Reordena los documentos basado en el criterio.

## \$match

Sirve para filtrar documentos que cumplan cierto criterio, de manera similar a lo que a lo que hacíamos con el método *find({<condiciones>})*.

```
// Trae los documentos con "name" Zachary
> db.Alumnos.aggregate([
  { $match: {
    name: 'Zachary'
  } }
])
```

## \$project

// Trae los documentos sólo con "name"

```
> db.Alumnos.aggregate([
  { $project: { _id: 0, name: 1 } }
])
```

// Trae los documentos con un nuevo campo "fullname"

```
> db.Alumnos.aggregate([
  { $project: {
    fullname: {$concat: ['$name', ' ', '$lastname']}
  } }
])
```

<https://docs.mongodb.com/manual/reference/operator/aggregation/>

## \$sort

```
> db.Alumnos.aggregate([
  { $project: { name: 1 } },
  { $sort: { name: 1 } }
])
```

# \$unwind

```
> db.Alumnos.aggregate([
  { $unwind: "$scores" }
])
```

```
/* 1 */
{
  "_id" : 0,
  "name" : "aimee",
  "lastname" : "Zank",
  "scores" : [
    {
      "score" : 1.46317973670502,
      "type" : "exam"
    },
    {
      "score" : 11.7827330995777,
      "type" : "quiz"
    },
    {
      "score" : 35.8740349954354,
      "type" : "homework"
    }
  ]
}
```

```
/* 1 */
{
  "_id" : 0,
  "name" : "aimee",
  "lastname" : "Zank",
  "scores" : {
    "score" : 1.46317973670502,
    "type" : "exam"
  }
}
/* 2 */
{
  "_id" : 0,
  "name" : "aimee",
  "lastname" : "Zank",
  "scores" : {
    "score" : 11.7827330995777,
    "type" : "quiz"
  }
}
/* 3 */
{
  "_id" : 0,
  "name" : "aimee",
  "lastname" : "Zank",
  "scores" : {
    "score" : 35.8740349954354,
    "type" : "homework"
  }
}
```

## \$group

<https://docs.mongodb.com/manual/reference/operator/aggregation/group/>

```
> db.Alumnos.aggregate([
  { $unwind: "$scores" },
  { $group: {
    _id: "$_id", // campo por el cual agrupo
    // nueva columna “promedio”
    promedio: { $avg: "$scores.score" },
    cantidad: { $sum: 1 }
  }}
])
```

Las nuevas columnas siempre deben llevar un valor “acumulado” como puede ser promedio, suma, máximo, etc.

## \$group (cont.)

```
> db.Alumnos.aggregate([
  { $unwind: "$scores" },
  { $group: {
    _id: "$name", // agrupo por nombre
    promedio: { $avg: "$scores.score" }
  }}
])
```

**¡Cuidado que puede no ser único!**

## \$group (cont.)

```
> db.Alumnos.aggregate([
  { $unwind: "$scores" },
  { $group: {
    _id: "$scores.type", // agrupo por tipo de examen
    promedio: { $avg: "$scores.score" }
  }}
])
```

En este caso se generarán 3 grupos (“homework”, “quiz” y “exam”)

## \$group (cont.)

```
> db.Alumnos.aggregate([
  { $unwind: "$scores" },
  { $group: {
    _id: null, // generará un solo grupo
    promedio: { $avg: "$scores.score" }
  }}
])
```

**En caso de agrupar por el valor “*null*” se utilizará un solo grupo con todos los documentos.**

# MongoDB University

<https://university.mongodb.com/courses/catalog>

The screenshot shows the MongoDB University website's course catalog. At the top, there's a navigation bar with icons for learning, developing, and innovating, followed by a message about the MongoDB.live series. The main header is "Online Course Catalog". Below it, three courses are listed in cards:

- M001: MongoDB Basics** (Introductory): An introductory course for setting up an Atlas cluster, querying data, and creating applications.
- M100: MongoDB for SQL Pros** (Introductory): A course designed for users with RDBMS and SQL knowledge, helping them learn MongoDB.
- M103: Basic Cluster Administration** (Introductory): A course on starting basic MongoDB deployments, including single mongod processes, replica sets, and sharded clusters.