

# BASES DE DATOS

db.collection.find({})

Clúster: grupo de servidores que almacenan sus datos  
"group of servers that store your data"

Miclustre: usuario: m001 - student

password: m001 - mongodb-basics

mongo "mongodb+srv://sandbox.u09nx.mongodb.net/"

-- username m001-student -- password m001 -  
mongodb-basics

\* JSON: JavaScript Object Notation

\* Para que un documento cumpla con el formato  
JSON debe empezar y acabar con {}

\* Separan cada clave y valor con -> :

\* Separan cada clave : valor con -> ,

\* Añadir comillas alrededor de las claves -> " "

(las claves también se conocen como campos en  
MongoDB)

\* Los datos de MongoDB se almacenan en BSON  
pero se ven en JSON

// Importar y exportar datos

JSON

mongoimport  
mongoexport

BSON

mongorestore  
mongodump

// Use show dbs and show collections for available namespace.

- \* `find()` returns a cursor with documents that match the find query.
- \* `count()` returns the number of documents that match the find query.
- \* `pretty()` formats the documents in the cursor
- \* `db.zips.find({ "": "" }).count()`

// Add documents \* every document must have a unique - id value

- \* `ObjectId()`
  - ↳ default value for the - id field unless otherwise specified.

example:

```
"_id": ObjectId("5ec5f1b710ca9222")
```

- \* El valor de ID subrayado sirve como identificador único para un documento en una colección y es un campo obligatorio para cada documento en MongoDB.
- \* `ObjectId()` is the default value for the "\_id" field unless otherwise specified

\* Insert multiple documents by using an array:  
Insertar varios documentos con ayuda de una matriz  
`db.collection.insert([ {<doc1>} , {<doc2>} ] )`

Use `{"ordered": false}` to disable the default ordered insert.

Collections and databases are created when las colecciones y bases de datos se crean cuando they are being used:  
utilizamos:

use tools followed by db.tools.insert({<trackets doc>}) creates the tools-tools namespace near el

\* Identical documents can exist in the same collection as long as their \_id values are different.  
MongoDB permite tener documentos idénticos en la colección, siempre que sus valores \_id sean diferentes.

\* Ver colecciones en db: `show collection`

\* Ver todas las bases de datos disponibles: `show dbs`

~~Update operators~~ ~~el unset elimina un campo en particular~~ ~~el unset campo1: {}~~

\* Increments field value by a specified amount  
Incrementa el valor del campo en una cantidad específica.

`{ "$inc": { "pop": 10, "field2": <increment value>, ... } }`

\* Sets field value to a new specified value.

Establece el valor del campo en un nuevo valor especificado.

`{ "$set": { "pop": 17630, "field2": <new value>, ... } }`

~~también se utiliza para agregar nuevos campos~~

\* Adds an element to an array field.

Agrega un elemento a un campo de matriz

`{ "$push": { <field1>: <value1>, ... } }`

~~push: { "scores": { "type": "extracredit", "score": 100 } } }~~

añadir

campo anulado

valor añadido al nuevo campo

field : campo

db. inspections. drop()

✓ Eliminar:

- \* db.<collection>.drop()) deleted the given collection  
elimina la colección  
\* deleteOne(), deleteMany() deletes documents that  
match a given query  
ejemplos: no se borrar la colección si está vacía.

Delete all the documents that have test field equal to 1

L D db.inspections.deleteMany ({"test": 1})

Delete one document that has test field equal to 3.

L D db.inspections.deleteOne ({"test": 3})

- ✓ Update all documents in the zips collection where the city field is equal to "HUDSON" by adding 10 to the current value of the "pop" field.

db.zips.updateMany ({"city": "HUDSON"}, {"\$inc": {"pop": 10}})

Corrección ejercicio 1:

1. array: dentro de un campo, varios valores

2.

3. insertOne: "One" en mayúscula

4. { "t-shirt number": { \$gt: 5 } }

✓ Comparison operators

\* \$eq = Equal to (igual que)

(Operadores de comandos)

\$neq = Not Equal (no igual que)

\* \$gt > Greater Than (mayor que)

\$lt < Less Than (menor que)

campo: { \$ne: value }

{campo: valor, \$lt: valor} } .count()

\$nor: [ \$ ] } .count()

\* \$gte ≥ Greater Than or Equal to (mayor o igual que)

\$lte ≤ Less Than or Equal to (menor o igual que)

{<field>: {<operator>: <value>} }

{"tipduracion": {"\$lte": 60}}

\$eq → is used as the default operator when an operator is not specified.

Equality se utiliza como operador predeterminado

db.orders.find({ "q": "60" })

## Conecitar:

1. Copiar el enlace desde el Atlas.

2. poner la contraseña

3. use "collection" (para entrar dentro de la colección)

4. db."tips" (para meterme dentro de la subcolección)

5. db.tips.find({ "pop": {"\$lt": 1.000 } }).count()

## Logic operators

\* \$and Match all of the specified query clauses

coincidir con todas las cláusulas de consulta especificadas

\* \$or At least one of the query clauses is matched

al menos una de las cláusulas de consulta coincide

\* \$nor Fail to match both given clauses → ni una ni otra

No coincide con las dos cláusulas dadas

\* \$not Negates the query requirement

Nega el requisito de consulta

△ of "\$operator": [ { clause1 }, { clause2 } ]

△ \$not: { clause } { \$or: [ { campo: valor }, { campo: valor } ] }

ejemplo = me pide que sea igual a:

sector = Home Improvement - 100

result = Out of business

= db.inspections.find({ "result": "Out of business", "sector": "Home Improvement", "sector": "Contractor-100" }).count()

AND

~~W~~ \$lookup Realiza una unión externa a una colección no fragmentada en la misma base de datos para filtrar los documentos de la colección. Une dos colecciones si tienen algún dato en común.

Accumulator ? database

```
db.collection.$lookup({  
    from: <colección a unirse>,  
    localField: <campo de los documentos de entrada>,  
    foreignField: <campo de los documentos de la colección "from">,  
    as: <campo de matriz de salida>  
})
```

~~W~~ Schema validation.

introducimos un esquema que puede definir reglas para validar inserciones y actualizaciones antes de escribir en la base de datos.

le damos ciertas normas a cada campo y si el documento no cumple esas normas, no se incluirá el documento dentro de la colección.

### Validation Level

Que documentos se validan?

Which documents get validate?

strict → All insert and updates

strict → Todos los documentos y actualizaciones

moderate → All insert and

moderate → Todas las inserciones y actualizaciones

updates to correct documents

actualizaciones que corrija los documentos

### Validation Action

What does it happen if validation fails?

Que sucede si falla la validación?

Error → Throw error and deny insert/update

Error → Lanza error y deniega inserción/actualización

warning → Log warning but proceed

Error → Registra advertencia pero procede

Expressive Query Operator Signifies that you are looking at the value of that field rather than the field itself.

\* `$expr` → allows the use of aggregation expressions  
permite el uso de expresiones de agrupación  
within the query language.  
dentro del lenguaje de consulta

`{ "$expr": { <expression> } }` Denotes an operator

Allows us to use variables and conditional statements  
Nos permite usar variables y declaraciones condicionales.

`{"$expr": {"$eq": [{"$stat station id": "A"}, {"$end station id": "B"}]}}`

\* Es necesario poner "`$`" delante del campo cuando usemos  
`"$expr"` podemos comparar valores dentro del mismo de

Which of the following ... more employees than the year were founded?

`db.companies.find()`

`{"$expr": {"$gt": [{"$number_of_employees": "$founded_year"}]}}`

`db.companies.find()`

`{"$expr": {"$lt": [{"$founded_year": "$number_of_employees"}]}}`

## Array Operators

\* `$push`: Allows us to add an element to an array.  
Nos permite añadir un elemento a un array  
o convertir un campo en un array si es que era otra cosa.

\* `$size`: devolverá todos los documentos donde el campo  
el que tiene 3 arrays dentro de 1 array  
de matriz especificado tiene exactamente la longitud dada.

`{<array field>: { "$size": <number> }}`

\* `$all`: devolverá un cursor con todos los documentos en los que  
el campo de matriz especificado contiene todos los elementos

`{<array field>: { "$all": <array> }}` que se estén  
indicando en la matriz.

\* `accommodates > 6`      Sunset Beach Lodge Review  
reviews > 50

`db...({$and: [{ "accommodates": { "$gt": 6 } }, { "review": { "$size": 50 } } ]})`

"property-type" "house" eindlyan "Changing table" comuna "amenities"

find({ "\$and": [ { "property-type": "House" }, { "amenities": "Changning Park" } ] })

Array Operators and Projections query a subdocument-like matrix companies.find({  
  "employees.firstName": "Helen",  
  "city": "Seattle"}).com  
db.collection.find({<query>}, {<projection>} )

\$: elemMatch (un array dentro de otra array)

`}<field>: {"$elemMatch": {<field>: <value>}}`

\* Busca todas las compañías que tienen oficinas en Seattle:

db. companies - find /

```
        } "offices": { "$elemMatch": { "city": "Seattle" } } }).count()
```

## Array Operators and Sub documents

("away · subway": "value" 4) · count(1)

\* busca cuántos viajes comienzan en las estaciones que están al oeste de la coordenada -74? sample traing . tip

db.tips.find()

```
{"start station location.coordinates": {"$lt": -74}}, count)
```

\* How many inspections from the sample training collection were conducted in the city of New York?

## 2b. inspections. find (

```
of "address.city": "NEW YORK" }).count() const pt = [
```

## Capítulo 5.

Framework \* Find all documents that have WiFi as one of the amenities. Only include price and address in the resulting cursor.

db.listingsAndReviews.find({ "amenities": "wifi" }, { "price": 1, "address": 1, "id": 0 }).pretty

con framework:

db.listingsAndReviews.aggregate([ { \$match: { "amenities": "wifi" } }, { \$project: { "price": 1, "address": 1, "\_id": 0 } } ])

\$group: An operator that takes the incoming stream of data, and siphons it into multiple distinct reservoirs sucesiva en multiples depósitos distintos.

\* What room type are present in the sample-airbn listingsAndReviews collection?

db.listingsAndReviews.aggregate(

[ { \$project: { "room\_type": 1, "\_id": 0 } }, { \$group: { "\_id": "\$room\_type" } } ]

Sort and Limit //

db.zips.find().sort({ "pop": 1 }).limit(1)

Sirven para filtrar resultados con el orden y cantidad que buscamos.

\* Which of the following commands will return the name and founding year for the oldest companies in the sample-training.companies collection?

db.companies.find(

{ "founding\_year": { \$ne: null } },

{ "name": 1, "founded\_year": 1 } ).sort({ "founded\_year": 1 }).limit(5)

db.tips.find().sort({ "birth\_year": -1 }).limit(1)

## Indexes

Index: es similar a un índice de un libro, donde tiene una lista alfabética de nombres y materias con referencias a los lugares donde ocurren.

single field index

```
db.tips.createIndex({ "birth year": 1 })
```

```
db.tips.createIndex({ "start station id": 1, "birth year": 1 })
```

\* No importa si el índice se crea en orden creciente o decreciente cuando es un índice simple de 1 solo campo.

compound index

## DATA MODELING v2

- La regla más importante en el modelado de datos con Mongo, es que los datos se almacenan de la forma en que se utilizan

\* a way to organize fields in a document to support your application performance and querying capabilities.

## Upset - Update or Insert?

```
db.collection.updateOne({query to locate}, {<update>})
```

Upset is a hybrid of update and insert, it should only be used when it is needed.  
Upset es un híbrido de actualización e inserción, y solo debe usarse cuando sea necesario.

```
db.collection.updateOne({query}, {<update>}, {"upsert": true})
```

\* La inserción ocurrirá si no hay documentos que coinciden con este criterio

\* La actualización ocurrirá si hay documentos que coinciden con los criterios de filtrado de la actualización