MONGODB

show dbs: Mostrar las bases de datos

db: Devuelve el nombre de la base de datos activa

use <nombre_db> : Cambiar a la base de datos

show collections • Muestra las colecciones de la DB activa

db.createCollection(,) • Crea una nueva colección • Al insertar un elemento, se crea automáticamente si no existe

db..drop() • Elimina la colección

READ:

https://www.mongodb.com/docs/mongodb-shell/crud/read/

db.<collection_name>.findOne() • Devuelve un elemento de la colección

db. <collection_name>.find() • Devuelve todos los elementos de la colección • Si hay más de 20, devuelve 20 y un puntero para recorrerlos it • Avanzar a los siguientes 20 elementos

db. <collection_name>.find().pretty() • Devuelve los elementos para que sea más fácil leerlos db. <collection_name>.find().count() • Devuelve el número de elementos

db. <collection_name>.find(query) • Se puede añadir una query para filtrar la búsqueda

• Ej: db.zips.find({"state":"AL"}).count()

INSERT:

https://www.mongodb.com/docs/mongodb-shell/crud/insert/

db. <collection_name>.insertOne() • Inserta el documento en la colección

db. <collection_name>.insertMany([,,...]) • Inserta múltiples documentos

UPDATE:

https://www.mongodb.com/docs/mongodb-shell/crud/update/

db. <collection_name>.updateOne(filter, update, options)

db. <collection_name>.updateMany(filter, update, options) • Actualiza el valor de un campo • filter: selection criteria • update: modificación a aplicar

db. <collection_name>.replaceOne(filter, replacement, options) • Reemplaza un documento por otro • replacement: nuevo documento

Operadores para update: • $set: Para cambiar un valor • $unset: Para eliminar un campo • $inc: Para incrementar un número • $push: Para añadir un elemento a un array • Más operadores

Ej: db.zips.updateOne({"zip":"12534"},{"$set":{"pop":17630}})

DELETE

db. <collection_name>.deleteOne(filter, options)

db. <collection_name>.deleteMany(filter, options) • Elimina los documentos que cumplan el filtro • filter: criterio de selección

COMPARADORES

• $eq: = (default)

• $ne: !=

• $gt: >

• $lt: <

• $gte: >=

• $lte: <=

• $and (default)

• $not

• $nor

• $or

Ejemplo:

Documentos cuya duración del viaje sea menor de 70 segundos y el tipo de usuario no sea subscriptor

```
db.trips.find({"tripduration": { "$lte" : 70 }, "usertype": { "$ne": "Subscriber" } }).pretty()
```

Documentos donde los aviones CR2 o A81 aterrizaron o despegaron del aeropuerto KZN

```
db.routes.find({ "$and": [
        { "$or" :[ { "dst_airport": "KZN" }, { "src_airport": "KZN" }] },
        { "$or" :[ { "airplane": "CR2" }, { "airplane": "A81" } ] }
]}).pretty()
```

MongoDB – EJ1

1. En sample_training.zips ¿Cuántas colecciones tienen menos de 1000 personas en el campo pop? (sol. 8065)

```
db.zips.find({ "pop": { $lt: 1000 } }).count()
```

2. En sample_training.trips ¿Cuál es la diferencia entre la gente que nació en 1998 y la que nació después de 1998? (sol. 6)

```
db.trips.count({ "birth year": { $eq: 1998 } }) - db.trips.count({ "birth year": { $gt: 1998 } })
```

3. En sample_training.routes ¿Cuántas rutas tienen al menos una parada? (sol. 11)

```
db.routes.count({ "stops": { $exists: true, $not: { $size: 0 } } })
```

4. En sample_training.inspections ¿Cuántos negocios tienen un resultado de inspección "Out of Business" y pertenecen al sector "Home Improvement Contractor - 100"? (sol. 4)

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

5. En sample_training.inspections ¿Cuántos documentos hay con fecha de inspección "Feb 20 2015" o "Feb 21 2015" y cuyo sector no sea "Cigarette Retail Dealer - 127"? (sol. 204)

```
db.inspections.count({
  "date": { $in: ["Feb 20 2015", "Feb 21 2015"] },
  "sector": { $ne: "Cigarette Retail Dealer - 127" }
})
```

Ej. 1: Número de documentos de sample_training.trips donde el viaje empieza y termia en la misma estación:

```
db.trips.find({ "$expr": { "$eq": [ "$end station id", "$start station id"] }}).count()
```

Ej. 2: Find all documents where the trip lasted longer than 1200 seconds, and started and ended at the same station:

```
db.trips.find({ "$expr":
        { "$and": [
                { "$gt": [ "$tripduration", 1200 ]},
                { "$eq": [ "$end station id", "$start station id" ]}
]}}).count()
```

$push: añadir un elemento a un array (crearlo si no existe)

```
db.students.insertOne({_id: 1, scores: [44, 78, 38, 80]})
db.students.updateOne({ _id: 1 }, {$push: {scores: 89}})
```

En sample_training.listingsAndReviews:

{"amenities": "Shampoo"} ◊ – Permite buscar en el array

{"amenities": ["Shampoo"]} ◊ – Busca exactamente ese array

"$all": [...] • Para buscar arrays que contengan al menos esos elementos • Ej.: db.listingsAndReviews.find({ "amenities": { "

$all": ["Internet", "Wifi", "Kitchen", "Heating",

"Family/kid friendly", "Washer", "Dryer", "Essentials"]

}})

"$size": number • Para especificar un tamaño exacto del array

• Ej.: db.listingsAndReviews.find({"amenities":{"$size":55}}).count()

"$elemMatch" • Usado con arrays, los proyecta sólo si tienen un elemento que cumpla el criterio

• Ej.: db.grades.find({"class_id": 431}, {"scores": {"$elemMatch": {"score": {"$gt": 85}}}})

"$regex" • Para buscar patrones en Strings usando expresiones regulares

• Ej.: db.companies.find({ "relationships.0.person.first_name": "Mark", "relationships.0.title": {"$regex": "CEO" } }, { "name": 1 })

MongoDB – EJ2

1. En sample_training.companies, ¿cuántas empresas tienen más empleados que el año en el que se fundaron? (sol. 324)

```
db.companies.count({
  $expr: { $gt: ["$number_of_employees", { $subtract: ["$founded_year", 1900] }] }
})
```

2. En sample_training.companies, ¿en cuántas empresas coinciden su permalink con su twitter_username? (sol. 1299)

```
db.companies.count({
  $expr: { $eq: ["$permalink", "$twitter_username"] }
})
```

3. En sample_airbnb.listingsAndReviews, ¿cuál es el nombre del alojamiento en el que pueden estar más de 6 personas alojadas y tiene exactamente 50 reviews? (sol. Sunset Beach Lodge Retreat)

```
db.listingsAndReviews.find({
  "accommodates": { $gt: 6 },
  "reviews": { $size: 50 }
}, { "name": 1, "_id": 0 }).pretty()
```

4. En sample_airbnb.listingsAndReviews, ¿cuántos documentos tienen el "property_type" "House" e incluyen "Changing table" como una de las "amenities"? (sol. 11)

db.listingsAndReviews.count({

"property_type": "House",

"amenities": "Changing table"

})

5. En sample_training.companies, ¿Cuántas empresas tienen oficinas en Seattle? (sol. 117)

db.companies.count({

"offices.city": "Seattle"

})

6. En sample_training.companies, haga una query que devuelva únicamente el nombre de las empresas que tengan exactamente 8 "funding_rounds"

db.companies.find({

"funding_rounds": { $size: 8 }

}, { "name": 1, "_id": 0 }).pretty()

7. En sample_training.trips, ¿cuántos viajes empiezan en estaciones que están al oeste de la longitud -74? (sol. 1928) Nota 1: Hacia el oeste la longitud decrece Nota 2: el formato es : [,]

db.trips.count({

"start station location": { $geoWithin: { $box: [[-180, -90], [-74, 90]] } }

})

8. En sample_training.inspections, ¿cuántas inspecciones se llevaron a cabo en la ciudad de "NEW YORK"? (sol. 18279)

db.inspections.count({ "city": "NEW YORK" })

9. En sample_airbnb.listingsAndReviews, haga una query que devuelva el nombre y la dirección de los alojamientos que tengan "Internet" como primer elemento de "amenities"

db.listingsAndReviews.find({

"amenities.0": "Internet"

}, { "name": 1, "address": 1, "_id": 0 }).pretty()

MongoDB – Aggregation

{$match: { <query> }} • Devuelve todos los documentos que cumplan la query • Las queries son equivalentes a las de lectura (los find)

• Ej.: db.listingsAndReviews.aggregate([{ "$match": { "amenities": "Wifi" } }])

{$project: { }} • Devuelve todos los documentos que cumplan la query • Las queries son equivalentes a las de lectura (los find) • Más información

• Ej.: db.listingsAndReviews.aggregate([

{"$match": {"amenities": "Wifi"}},

{"$project": {"price": 1, "address": 1}}])

{$group:{

_id: , // Group By Expression

<field1>: { <accumulator1> : <expression1>}, ... }} • Agrupa todos los elementos que sean iguales • : – Operación a realizar – Ej.: $count, $max, $min, $sum...

$group

• Ej.: db.listingsAndReviews.aggregate([

{"$project": {"address": 1, "_id": 0 }},

{"$group": {"_id": "$address.country" }}])

db.listingsAndReviews.aggregate([

{"$project": {"address": 1, "_id": 0 }},

{"$group": {"_id": "$address.country", "count": {"$sum": 1 } } }])

MongoDB – EJ3

1. En sample_airbnb.listingsAndReviews, ¿qué "room types" existen?

db.listingsAndReviews.distinct("room_type")

2. En sample_training.companies, haga una query que devuelva el nombre y el año en el que se fundaron las 5 compañías más antiguas.

db.companies.find({}, { "name": 1, "founded_year": 1, "_id": 0 }).sort({ "founded_year": 1 }).limit(5)

3. En sample_training.trips, ¿en qué año nació el ciclista más joven? (sol. 1999)

db.trips.aggregate([

{ $group: { _id: null, youngestBirthYear: { $min: "$birth year" } } }

])

SIMULACRO

Problema 2

Haciendo uso de mongosh complete el archivo problema2.txt con las operaciones que se piden en los siguientes apartados. Para cada una indique únicamente la instrucción realizada, no hace falta indicar la solución. De forma que la respuesta del apartado 1 quede por ejemplo como se muestra a continuación:

Apartado 1

Instrucción

Tenga en cuenta que cada apartado se responde con una única consulta y que esta tiene que seguir siendo válida si se añaden o eliminan documentos de la colección.

"Hechos con colección simulacro"

Apartado 1.

En la colección **`listingAndReviews`** indique el/los nombre(s) del alojamiento con más reviews.

db.simulacro.find({}, {name: 1, number_of_reviews: 1, _id: 0}).sort({number_of_reviews: -1}).limit(1)

1. **`db.simulacro.find({})`**: Busca todos los documentos en la colección **`simulacro`**.
2. **`{name: 1, number_of_reviews: 1, _id: 0}`**: Proyecta (selecciona) solo los campos **`name`** y **`number_of_reviews`** en los resultados, y excluye el campo **`_id`**.
3. **`.sort({number_of_reviews: -1})`**: Ordena los resultados en orden descendente según el campo **`number_of_reviews`**.
4. **`.limit(1)`**: Limita los resultados a solo un documento.

Apartado 2.

En la colección **`listingAndReviews`** indique el/los nombre(s) del alojamiento con más amenities.

db.simulacro.find({}, {name: 1, amenities: 1, _id: 0}).sort({amenities: -1}).limit(1)

1. **`db.simulacro.find({})`**: Busca todos los documentos en la colección **`simulacro`**.
2. **`{name: 1, amenities: 1, _id: 0}`**: Proyecta (selecciona) solo los campos **`name`** y **`amenities`** en los resultados, y excluye el campo **`_id`**.
3. **`.sort({amenities: -1})`**: Ordena los resultados en orden descendente según el campo **`amenities`** (asume que **`amenities`** es un array y el orden se basa en el tamaño del array).
4. **`.limit(1)`**: Limita los resultados a solo un documento.

Apartado 3.

En la colección `listingAndReviews` indique para cada tipo de `property_type` el número de alojamientos de ese tipo.

```
db.simulacro.aggregate([
  { $group: { _id: "$property_type", count: { $sum: 1 } } }
])
```

1. `db.simulacro.aggregate()`: Utiliza el método de agregación para procesar datos en la colección `simulacro`.
2. `[ { $group: { _id: "$property_type", count: { $sum: 1 } } } ]`: Agrupa los documentos por el campo `property_type` y cuenta el número de documentos en cada grupo.
   - `$group`: Especifica la operación de agrupamiento.
   - `_id: "$property_type"`: Agrupa por el valor del campo `property_type`.
   - `count: { $sum: 1 }`: Cuenta el número de documentos en cada grupo incrementando en 1 para cada documento.

Apartado 4.

En la colección `listingAndReviews` indique el número de alojamientos que tienen 2, 3, 4 o 5 `beds`.

```
db.simulacro.countDocuments({ beds: { $in: [2, 3, 4, 5] } })
```

1. `db.simulacro.countDocuments({ beds: { $in: [2, 3, 4, 5] } })`: Cuenta el número de documentos en la colección `simulacro` que tienen un campo `beds` con valores que están en el array `[2, 3, 4, 5]`.
   - `beds: { $in: [2, 3, 4, 5] }`: Selecciona los documentos donde el campo `beds` tiene un valor que está en el array `[2, 3, 4, 5]`.