ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

Laboratorio de: ANALÍTICA DE DATOS – BIG DATA

Tema: Laboratorio de la clase 5 sobre comandos de Mongo DB

Estudiante: Danny Sebastián Díaz Padilla

Objetivos:

- Leer una base de datos JSON con Mongo DB.
- Practicar los comandos propuestos en clase.
- Evaluar diez nuevos comandos no vistos en clase por medio de la consola de MongoDB.

Marco teórico:

Funciones

- **Pretty**

El método pretty () se usa principalmente para mostrar el resultado en un formato más fácil de leer. La sintaxis del método pretty () es la siguiente:

db.collection.find (). pretty ()

- **Data size**

Muestra el tamaño en bytes de la colección. La compresión de datos no afecta este valor. La sintaxis del método dataSize () es la siguiente:

db.collection.find (). dataSize ()

- **Storage Size**

Devuelve la cantidad total de almacenamiento asignado a esta colección para el almacenamiento de documentos. Si los datos de recopilación están comprimidos el tamaño de almacenamiento refleja el tamaño comprimido y puede ser menor que el valor devuelto por db.collection.dataSize (). Su sintaxis es la siguiente:

db.collection.find (). storageSize ()

- **Find one**

Devuelve un documento que satisface los criterios de consulta especificados en la colección o vista. Si varios documentos satisfacen la consulta, este método devuelve el primer documento de acuerdo con el orden natural que refleja el orden de los documentos en el disco.

db.collection.find (). findOne ()

Find

Selecciona documentos en una colección o vista y devuelve un cursor a los documentos seleccionados.

db.collection.find (). find({"tipo_delito":"Robo (Con violencia)"})

Count

Devuelve el recuento de documentos que coincidirían con una consulta find () para la colección o vista. El método db.collection.count () no realiza la operación find () sino que cuenta y devuelve el número de resultados que coinciden con una consulta.

*db.collection.count(*query, options*)*

Distinct

Encuentra los valores distintos para un campo específico en una única colección o vista y devuelve los resultados en una matriz.

*db.collection.distinct(*field, query, options*)*

Create Index

Crea un índice dentro de las colecciones.

db.collection.createIndex(*keys*, *options*)

GeoSearch

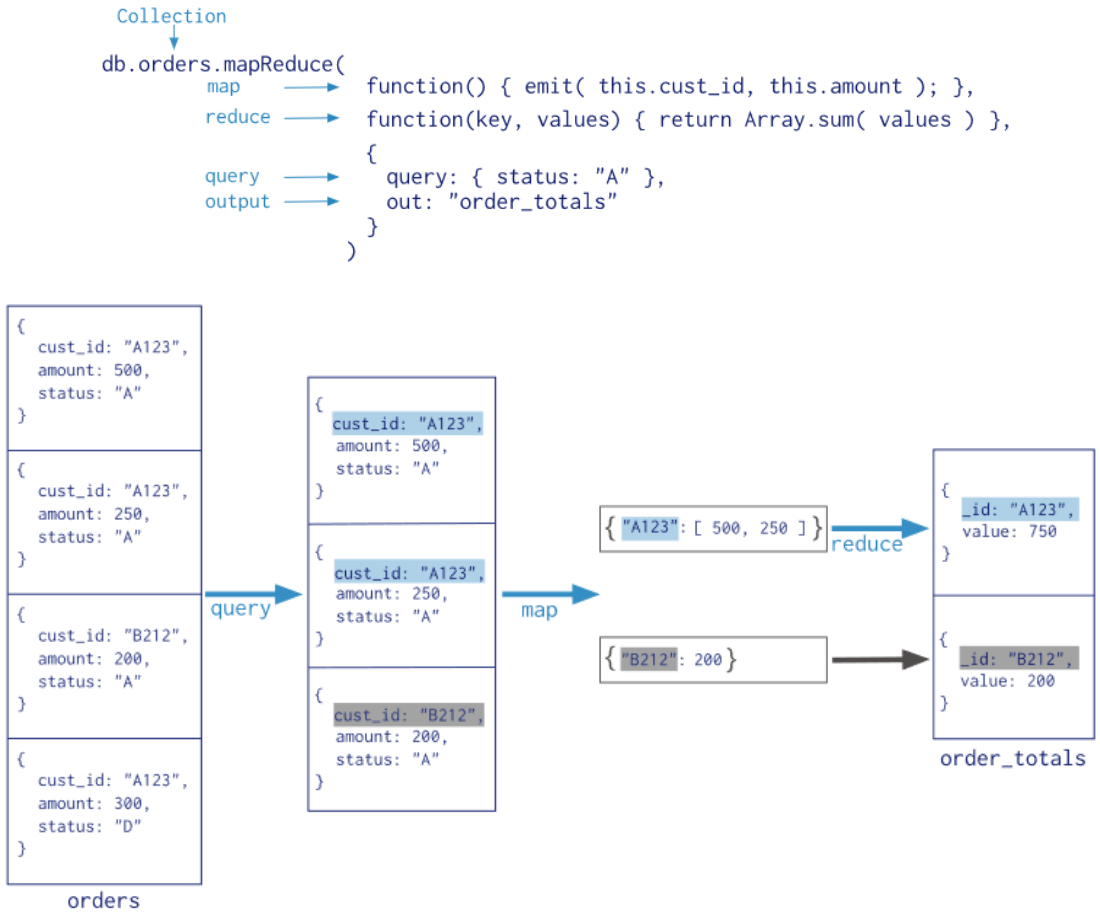
El comando geoSearch proporciona una interfaz para la funcionalidad de índice de pajar de MongoDB. Estos índices son útiles para devolver resultados basados en coordenadas de ubicación después de recopilar resultados basados en alguna otra consulta .
Sintaxis:

```
db.runCommand({
  geoSearch : "places",
  near: [ -73.9667, 40.78 ],
  maxDistance : 6,
  search : { type : "restaurant" },
  limit : 30
})
```

MapReduce

Map-reduce es un paradigma de procesamiento de datos para condensar grandes volúmenes de datos en resultados agregados útiles. Para las operaciones de reducción de mapas, MongoDB proporciona el comando de base de datos mapReduce.

Sintaxis y forma de trabajo:



Atributos

Sum: $SUM

Calcula y devuelve la suma de valores numéricos. $ sum ignora los valores no numéricos.

{ $sum: <expression> }

Average: $AVG

Calcula y devuelve el promedio de los valores numéricos. $ avg ignora los valores no numéricos.

{ $avg: <expression> }

- **Greater than or equal to: $gte**

$gte compara tanto el valor como el tipo, utilizando el orden de comparación BSON especificado para valores de diferentes tipos.

{ $gte: [<expression1>, <expression2>] }

- **Less than: $lt**

$ lt selecciona los documentos donde el valor del campo es menor que (es decir, <) el valor especificado.

{field: {$lt: value}

- **Equal tan: $eq**

$eq compara tanto el valor como el tipo, utilizando el orden de comparación BSON especificado para valores de diferentes tipos.

{ $eq: [<expression1>, <expression2>] }

- **Group: $group**

Agrupa documentos de entrada por la expresión _id especificada y para cada agrupación distinta, genera un documento. El campo _id de cada documento de salida contiene el grupo único por valor. Los documentos de salida también pueden contener campos calculados que contienen los valores de alguna expresión de acumulador.

db.sales.aggregate([{ $group : { _id : "$item" } }])

- **OR (Operación lógica): $or**

El operador $or realiza una operación OR lógica en una matriz de dos o más <expresiones> y selecciona los documentos que satisfacen al menos una de las <expresiones>. El $or tiene la siguiente sintaxis:

{ $or: [{<expression1>},{ <expression2> },..., { <expressionN> }] }

- **Sort: $sort**

Ordena los datos.

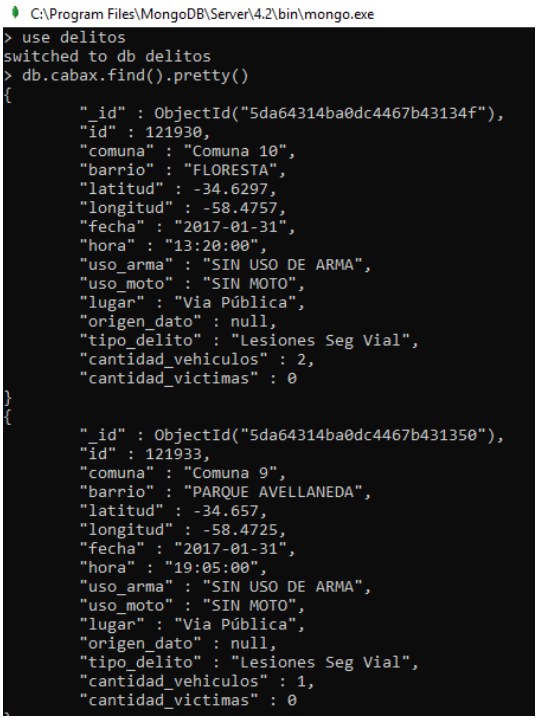
{ $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }

Desarrollo de la práctica:

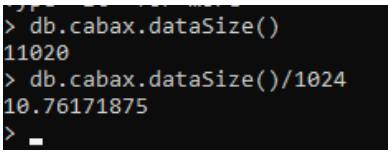
A. Trabajo en clase: nuevos comandos sobre MongoDB

Listamos todos los documentos de la base de datos agregada desde JSON para ver de que trata la colección:

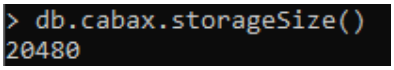
db.cavax.find().pretty()



Es posible ver el tamaño de la colección en bytes usando: **db.cabax.dataSize(),** asi mismo podemos realizar una operación sobre esa instrucción. En el ejemplo de abajo se calcula el peso en KB



Para mostrar el total de documentos dentro de la colección se usa: **db.cabax.storageSize()**

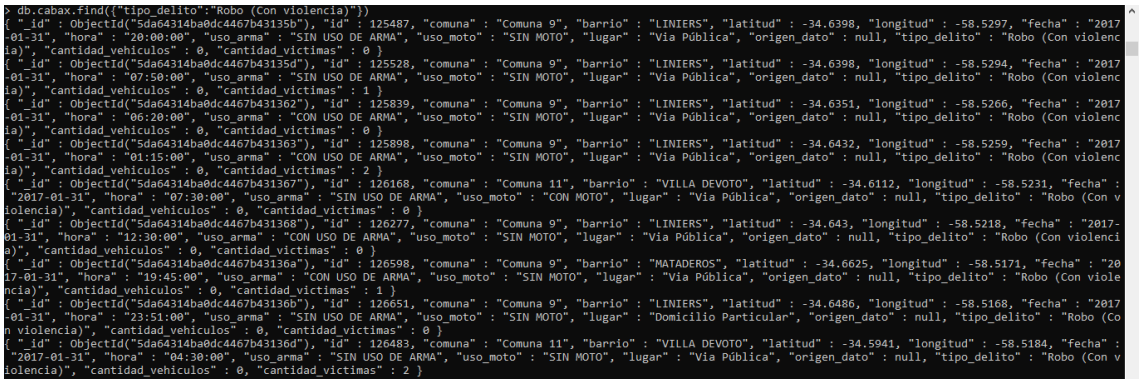


Para mostrar el primer elemento de la colección se usa: **db.cabax.findOne()**

```
> db.cabax.findOne()
{
        "_id" : ObjectId("5da64314ba0dc4467b43134f"),
        "id" : 121930,
        "comuna" : "Comuna 10",
        "barrio" : "FLORESTA",
        "latitud" : -34.6297,
        "longitud" : -58.4757,
        "fecha" : "2017-01-31",
        "hora" : "13:20:00",
        "uso_arma" : "SIN USO DE ARMA",
        "uso_moto" : "SIN MOTO",
        "lugar" : "Vía Pública",
        "origen_dato" : null,
        "tipo_delito" : "Lesiones Seg Vial",
        "cantidad_vehiculos" : 2,
        "cantidad_victimas" : 0
}
>
```

Búsqueda con filtro de Delitos en argentina que fueron del modo: Robo (Con violencia)

db.cabax.find({"tipo_delito":"Robo (Con violencia)"})



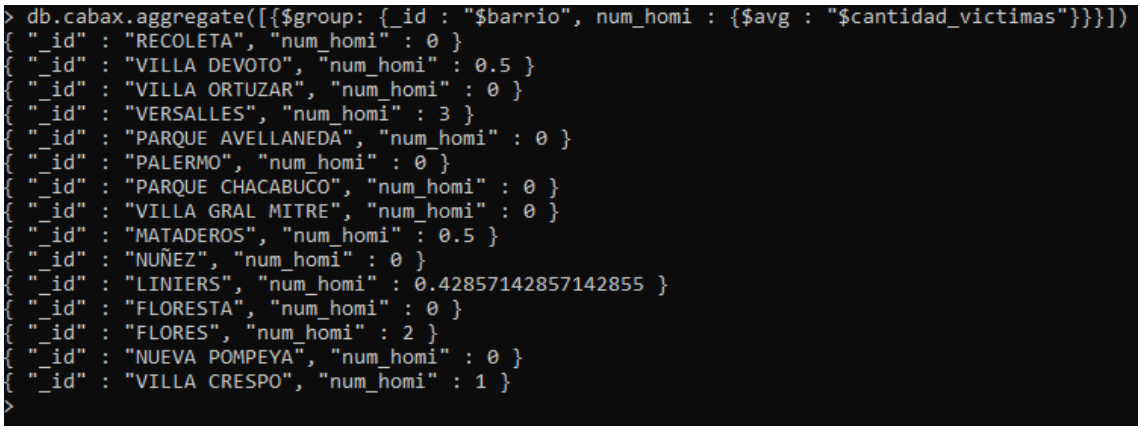
Sumar los registros de la cantidad de víctimas por barrio:

db.cabax.aggregate([{$group: {_id : "$barrio", num_homi : {$sum : "$cantidad_victimas"}}}])

```
> db.cabax.aggregate([{$group: {_id : "$barrio", num_homi : {$sum : "$cantidad_victimas"}}}])
{ "_id" : "NUEVA POMPEYA", "num_homi" : 0 }
{ "_id" : "VILLA CRESPO", "num_homi" : 1 }
{ "_id" : "MATADEROS", "num_homi" : 1 }
{ "_id" : "VILLA GRAL MITRE", "num_homi" : 0 }
{ "_id" : "FLORESTA", "num_homi" : 0 }
{ "_id" : "LINIERS", "num_homi" : 6 }
{ "_id" : "NUÑEZ", "num_homi" : 0 }
{ "_id" : "FLORES", "num_homi" : 2 }
{ "_id" : "PALERMO", "num_homi" : 0 }
{ "_id" : "PARQUE AVELLANEDA", "num_homi" : 0 }
{ "_id" : "PARQUE CHACABUCO", "num_homi" : 0 }
{ "_id" : "VILLA ORTUZAR", "num_homi" : 0 }
{ "_id" : "RECOLETA", "num_homi" : 0 }
{ "_id" : "VILLA DEVOTO", "num_homi" : 2 }
{ "_id" : "VERSALLES", "num_homi" : 3 }
>
```

Operación de la Media de la Cantidad de víctimas por barrio:

db.**cabax.aggregate([{$group: {_id : "$barrio", num_homi : {$avg : "$cantidad_victimas"}}}])**



B. Revisar y aplicar 10 comandos en el dataset delitosar.json, que no se hayan visto en clase.

a. Describir y explicar los comandos utilizados. En el marco teórico.
b. Aplicar y analizar los delitos del dataset
c. Mostrar resultados

1. Count

Comando: db.cabax.count({lugar:"Via Pública"})
Descripción: Se utiliza para contar los delitos que han ocurrido en la vía pública.
Uso:

```
> db.cabax.count({lugar:"Via Pública"})
31
```

2. Distinct

Comando: db.runCommand({distinct: “cabax”, key: “comuna”})
Descripción: Se utiliza para seleccionar todos los datos distintos de una clave en específico.
Uso:

```
> db.runCommand ( { distinct: "cabax", key: "comuna" } )
{
        "values" : [
                "Comuna 10",
                "Comuna 9",
                "Comuna 14",
                "Comuna 15",
                "Comuna 11",
                "Comuna 7",
                "Comuna 2",
                "Comuna 13",
                "Comuna 4"
        ],
        "ok" : 1
}
```

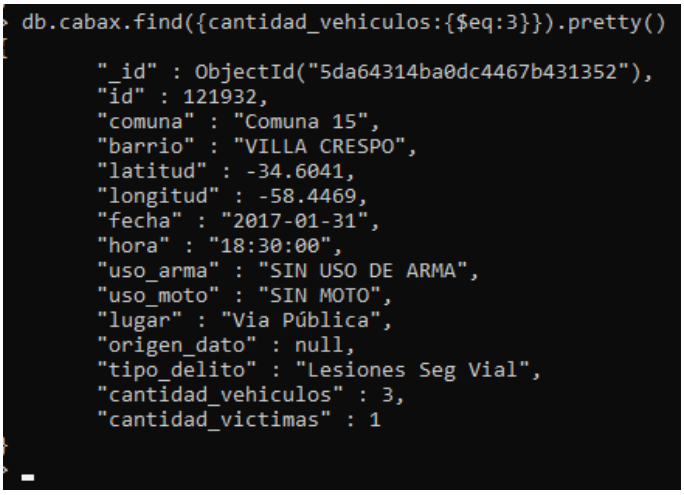
3. Greater than or equal to

Comando: db.cabax.find({cantidad_victimas:{$gte:2}}).pretty()
Descripción: Encuentra todos los delitos cuyo número de víctimas fue mayor o igual a 2.
Uso:

```
> db.cabax.find({cantidad_victimas:{$gte:2}}).pretty()
{
        "_id" : ObjectId("5da64314ba0dc4467b431357"),
        "id" : 121938,
        "comuna" : "Comuna 7",
        "barrio" : "FLORES",
        "latitud" : -34.6395,
        "longitud" : -58.4586,
        "fecha" : "2017-01-31",
        "hora" : "20:00:00",
        "uso_arma" : "SIN USO DE ARMA",
        "uso_moto" : "SIN MOTO",
        "lugar" : "Via Pública",
        "origen_dato" : null,
        "tipo_delito" : "Lesiones Seg Vial",
        "cantidad_vehiculos" : 2,
        "cantidad_victimas" : 2
}
{
        "_id" : ObjectId("5da64314ba0dc4467b431363"),
        "id" : 125898,
        "comuna" : "Comuna 9",
        "barrio" : "LINIERS",
        "latitud" : -34.6432,
        "longitud" : -58.5259,
        "fecha" : "2017-01-31",
        "hora" : "01:15:00",
        "uso_arma" : "CON USO DE ARMA",
        "uso_moto" : "SIN MOTO",
        "lugar" : "Via Pública",
        "origen_dato" : null,
        "tipo_delito" : "Robo (Con violencia)",
        "cantidad_vehiculos" : 0,
        "cantidad_victimas" : 2
```

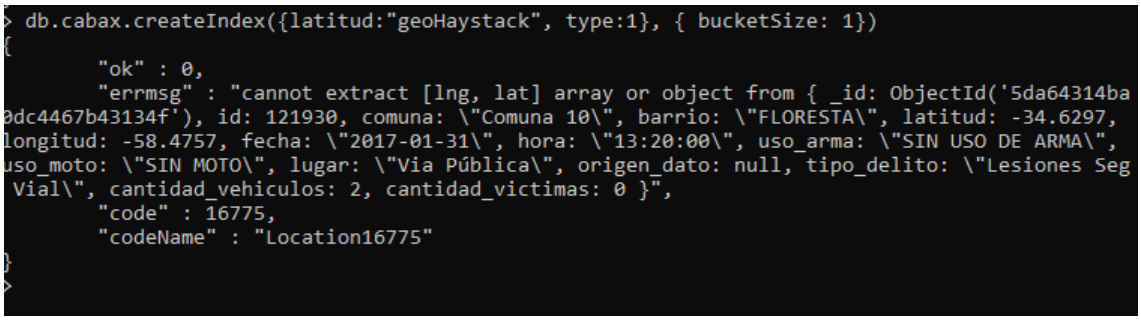
4. Less than

Comando: db.cabax.find({cantidad_vehiculos:{$eq:3}}).pretty()
Descripción: El comando busca los delitos en donde se hayan involucrado 3 vehículos exactamente.
Uso:



5. Create Index

Comando: db.cabax.createIndex({latitud:"geoHaystack", type:1}, { bucketSize: 1})
Descripción: Crea un índice especializado para utilizar una búsqueda geográfica
Uso:
En este caso el comando no fue exitoso porque se necesita [longitud, latitud] y esos datos en los documentos de la colección **cabax** estan separados



6. GeoSearch

Comando: db.runCommand({ geoSearch : "cabax", near: [-34.6297, -58.4757], maxDistance : 6, search : { cantidad_victimas : 1 }, limit : 10 })
Descripción: Busca los delitos alrededor del punto [-34.6297, -58.4757] cuyo número de víctimas es 1.
Uso:
De la mano con el comando anterior, este comando necesita crear un índice “geoHaystack”, el cual no se pudo agregar en la parte anterior por la estructura de los documentos

```
> db.runCommand({    geoSearch : "cabax",    near: [ -34.6297, -58.4757 ],    maxDistance : 6,
    search : { cantidad_victimas : 1 },    limit : 10 })
{ "ok" : 0, "errmsg" : "no geoSearch index" }
>
```

7. Sort

Comando: db.cabax.find({comuna: "Comuna 7"}).sort({cantidad_victimas: 1}).pretty()
Descripción: El comando se utiliza para ordenar los resultados en orden ascendente.
Uso:


```
> db.cabax.find({comuna: "Comuna 7"}).sort({cantidad_victimas: 1}).pretty()
{
        "_id" : ObjectId("5da64314ba0dc4467b431354"),
        "id" : 121936,
        "comuna" : "Comuna 7",
        "barrio" : "PARQUE CHACABUCO",
        "latitud" : -34.6311,
        "longitud" : -58.438,
        "fecha" : "2017-01-31",
        "hora" : "11:30:00",
        "uso_arma" : "SIN USO DE ARMA",
        "uso_moto" : "SIN MOTO",
        "lugar" : "Via Pública",
        "origen_dato" : null,
        "tipo_delito" : "Lesiones Seg Vial",
        "cantidad_vehiculos" : 1,
        "cantidad_victimas" : 0
}
{
        "_id" : ObjectId("5da64314ba0dc4467b431357"),
        "id" : 121938,
        "comuna" : "Comuna 7",
        "barrio" : "FLORES",
        "latitud" : -34.6395,
        "longitud" : -58.4586,
        "fecha" : "2017-01-31",
        "hora" : "20:00:00",
        "uso_arma" : "SIN USO DE ARMA",
        "uso_moto" : "SIN MOTO",
        "lugar" : "Via Pública",
        "origen_dato" : null,
        "tipo_delito" : "Lesiones Seg Vial",
        "cantidad_vehiculos" : 2,
        "cantidad_victimas" : 2
}
```

8. Group

Comando: db.cabax.aggregate([{ $group : { _id : "$comuna", count:{$sum:1} } }])
Descripción: Cuenta la cantidad de delitos u los agrupa por comuna.
Uso:

```
> db.cabax.aggregate( [ { $group : { _id : "$comuna", count:{$sum:1} } } ] )
{ "_id" : "Comuna 10", "count" : 3 }
{ "_id" : "Comuna 14", "count" : 1 }
{ "_id" : "Comuna 15", "count" : 2 }
{ "_id" : "Comuna 7", "count" : 2 }
{ "_id" : "Comuna 2", "count" : 1 }
{ "_id" : "Comuna 11", "count" : 5 }
{ "_id" : "Comuna 9", "count" : 18 }
{ "_id" : "Comuna 4", "count" : 1 }
{ "_id" : "Comuna 13", "count" : 1 }
```

9. OR (Operación lógica)

Comando: db.cabax.find({ $or: [{victimas: {$gte: 3}}, {comuna: {$eq: "Comuna 10"}}]}).pretty()
Descripción: Selecciona los delitos que tengan víctimas iguales o mayores atres O que hayan ocurrido en la comuna 10.
Uso:

```
db.cabax.find( { $or: [{victimas: {$gte: 3}}, {comuna: {$eq: "Comuna 10"}}]} ).pretty()

        "_id" : ObjectId("5da64314ba0dc4467b43134f"),
        "id" : 121930,
        "comuna" : "Comuna 10",
        "barrio" : "FLORESTA",
        "latitud" : -34.6297,
        "longitud" : -58.4757,
        "fecha" : "2017-01-31",
        "hora" : "13:20:00",
        "uso_arma" : "SIN USO DE ARMA",
        "uso_moto" : "SIN MOTO",
        "lugar" : "Via Pública",
        "origen_dato" : null,
        "tipo_delito" : "Lesiones Seg Vial",
        "cantidad_vehiculos" : 2,
        "cantidad_victimas" : 0


        "_id" : ObjectId("5da64314ba0dc4467b431356"),
        "id" : 121935,
        "comuna" : "Comuna 10",
        "barrio" : "FLORESTA",
        "latitud" : -34.62,
        "longitud" : -58.4924,
        "fecha" : "2017-01-31",
        "hora" : "00:15:00",
        "uso_arma" : "SIN USO DE ARMA",
        "uso_moto" : "SIN MOTO",
        "lugar" : "Via Pública",
        "origen_dato" : null,
        "tipo_delito" : "Lesiones Seg Vial",
        "cantidad_vehiculos" : 2,
        "cantidad_victimas" : 0


        "_id" : ObjectId("5da64314ba0dc4467b431365"),
        "id" : 126114,
        "comuna" : "Comuna 10",
        "barrio" : "VERSALLES",
        "latitud" : -34.6299,
        "longitud" : -58.5236,
        "fecha" : "2017-01-31",
```

10. MapReduce

Comando:

```
db.runCommand({
   mapReduce: "cabax",
   map: function(){
        emit("MAP: " + this.comuna, (this.cantidad_victimas+this.cantidad_vehiculos)/2 )
   },
   reduce: function(key, values){
        return Array.sum(values);
   },
   out: {inline:1}
})
```

Descripción: Obtenemos la media entre la suma de victimas y vehículos robados por cada delito (MAP) y luego sumamos todos los valores generados en el vector reduciéndolo a un valor escalar (REDUCE).

Uso:

```
> db.runCommand({ mapReduce: "cabax", map: function(){     emit("
MAP: " + this.comuna, (this.cantidad_victimas+this.cantidad_vehic
ulos)/2) }, reduce: function(key, values){ return Array.sum(value
s); }, out: {inline:1} })
{
        "results" : [
                {
                        "_id" : "MAP: Comuna 10",
                        "value" : 3.5
                },
                {
                        "_id" : "MAP: Comuna 11",
                        "value" : 1.5
                },
                {
                        "_id" : "MAP: Comuna 13",
                        "value" : 1
                },
                {
                        "_id" : "MAP: Comuna 14",
                        "value" : 0.5
                },
                {
                        "_id" : "MAP: Comuna 15",
                        "value" : 3
                },
                {
                        "_id" : "MAP: Comuna 2",
                        "value" : 2
                },
                {
                        "_id" : "MAP: Comuna 4",
                        "value" : 1
                },
                {
                        "_id" : "MAP: Comuna 7",
                        "value" : 2.5
                },
                {
                        "_id" : "MAP: Comuna 9",
                        "value" : 6
                }
        ],
        "timeMillis" : 35,
        "counts" : {
                "input" : 34,
                "emit" : 34,
                "reduce" : 5,
                "output" : 9
        },
        "ok" : 1
```

Análisis de resultados:

Todos los comandos fueron ejecutados exitosamente excepto GeoSearch. Debido a que necesita un índice especial "geoHaystack", el cual no se pudo agregar por la estructura de los documentos. Se requería un arreglo con [latitud, longitud], sin embargo, los delitos tenían esos valores separados.

Conclusiones y recomendaciones:

- De la practica: Se puede realizar analogías SQL hacia el lenguaje de consulta de Mongo DB.
- De la practica: MapReduce es una función muy útil con grandes volúmenes de datos y sobre todo si se ejecuta en un sistema distribuido.
- De los objetivos: Se pudo leer una base de datos JSON con Mongo DB.
- De los objetivos: Se practicó los comandos propuestos en clase.
- De los objetivos: Se evaluaron diez nuevos comandos no vistos en clase por medio de la consola de MongoDB. Pero uno no era compatible con la estructura del documento.
- Se recomienda utilizar la interfaz gráfica de MongoDB solo para inspeccionar los datos. La consola es mucho más versátil.
- Se recomienda refactorizar la estructura de los documentos para que encajen con las necesidades del comando GeoSearch

Bibliografía

MongoDB. (s.f.). *MongoDB*. Obtenido de MongoDB Web site:
https://docs.mongodb.com/manual