Guía # 7 – MongoDB

(Agregación Parte 1)

16.- CONTAR

Ejemplo 1: Total de datos de la colección estudiantes

```
> db.estudiantes.find( {} ).count() ;
> db.estudiantes.countDocuments() ;
```

Ejemplo 2: Total de damas de la colección estudiantes

```
> db.estudiantes.find( { genero:{$eq:"f"} } ).count() ;
> db.estudiantes.countDocuments( { genero:{$eq:"f"} } ) ;
```

Ejemplo 3: Total de damas con edad de 24 o 37 años

```
> db.estudiantes.countDocuments(
    {
        $and:[
                { genero: {$eq:"f"} } ,
                {
                    $or:[
                        { edad :{$eq:37} } , { edad :{$eq:24} }
                    ]
                }
        ]
    }
) ;
```

Ejemplo 4: Determinar la cantidad de estudiantes que programan en: "Python" o "Cobol" o "Java"

A.- Usando $in

```
> db.estudiantes.countDocuments(
      {
            programa:{
                  $in: ["Python",   "Cobol",   "java"]
            }
      }
) ;
```

B.- Usando $or

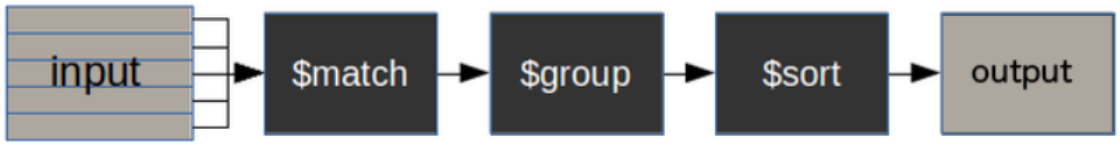
```
> db.estudiantes.countDocuments(
      {

            $or:[
                  { programa: "Python"  } ,
                  { programa: "Cobol"   } ,
                  { programa: "Java"    }
            ]
      }
) ;
```

17.- AGRECACIÓN

Cuando empiece a trabajar con MongoDB, normalmente utilizará la función find() para una amplia gama de consultas. Sin embargo, en cuanto sus consultas sean más avanzadas, necesitará saber más sobre la agregación de MongoDB.

La agregación es una forma de procesar un gran número de documentos de una colección haciéndolos pasar por distintas etapas.
Donde cada etapa constituye lo que se conoce como "pipeline **(tuberías)**", y consiste en un paso donde se filtran, ordenan, agrupan, remodelan o modifican los documentos que pasan por el pipeline (**tubo**).



La estructura para ejecutar un comando de agregación es la siguiente:

```
db.collection.aggregate(
    [
        { <etapa1 > } ,
        { <etapa2 > } ,
        *
        *
        { <etapaN > }
    ]
) ;
```

17.1.- $match: { } ---> Igual / find()

Este operador funciona de forma similar al método find() que ya vimos. Básicamente nos permite filtrar documentos de la colección según la consulta que pasemos como parámetro.

Se pueden usar los operadores:

- Relacionales:
  - Igualdad: { campo: valor } / { campo: {$eq: valor}}
  - Comparación:
    - $gt (mayor que) / $gte(mayor o igual que),
    - $lt (menor que) / $lte (menor o igual que),
    - $ne (no igual).
- Lógicos: $and, $or, $not.
- Elementos de Array: $in, $nin.
- Otros: $exists, $regex, etc..

```
> db.estudiantes.aggregate(
    [
        {
            $match: {
                <pregunta1> ,
                <pregunta2> ,
                *  ,
                <preguntaN>
            }
        }
    ]
) ;
```

Sintaxis

Ejemplo 1: Extraer todos los documentos dende este el nombre Zila

```
> db.estudiantes.find( { "nombre": {$eq:"Zila"} } ).toArray() ;
```

```
> db.estudiantes.aggregate(
    [
        {
            $match : {
                "nombre" : "Zila"
            }
        }
    ]
) ;
```

```
> db.estudiantes.aggregate(
    [
        {
            $match : {
                "nombre" : { $eq: "Zila" }
            }
        }
    ]
) ;
```

Ejemplo 2 ($AND) : Mostrar las Damas con Edad >= 50

```
> db.estudiantes.find(
        {
                $and:[
                        { "genero":{$eq:"f"}   } ,
                        { "edad":{$gte:50}   }
                ]
        }
).toArray() ;


> db.estudiantes.find(
        {
                "genero":{$eq:"f"} ,
                "edad":{$gte:50}
        } ,
        {
                _id:1 ,
                nombre: 1 ,
                genero: 1 ,
                edad: 1
        }
) ;
```

```
> db.estudiantes.aggregate(
    [
        {
            $match : {
                "genero":{$eq:"f"} ,
                "edad":{$gte:50}
            }
        } ,
        {
            $project: {  // Proyectar (Campos a mostrar)
                _id: 1,
                nombre: 1,
                genero: 1,
                edad: 1
            }
        }
    ]
) ;
```

Ejemplo 3 ($OR): Mostrar las personas con edad 24 o 37 años

```
> db.estudiantes.aggregate(
    [
        {
            $match : {
                $or:[
                    { edad : {$eq:24} } ,
                    { edad : {$eq:37} }
                ]

            }
        } ,
        {
            $project: { // Proyectar (Campos a mostrar)
                _id: 1,
                nombre: 1,
                genero: 1,
                edad: 1
            }
        }
    ]
) ;
```

Ejemplo 4 ($AND - $OR): Mostrar las damas con 24 o 37 años

```
> db.estudiantes.aggregate(
    [
        {
            $match : {
                "genero":{$eq:"f"} ,
                $or:[
                    { edad : {$eq:24} } ,
                    { edad : {$eq:37} }
                ]

            }
        } ,
        {
            $project: { // Proyectar (Campos a mostrar)
                _id: 1,
                nombre: 1,
                genero: 1,
                edad: 1
            }
        }
    ]
) ;
```

17.2.- { $count: } ---> Contar

Ejemplo 1: Total de estudiantes con el apellido Capacho

```
> db.estudiantes.find(
        {
                Apellido:     'Capacho'
        }
).count() ;

> db.estudiantes.aggregate(
        [
                {
                        $match : {
                                apellido :   "Capacho"

                        }
                } ,
                {
                        $count:"Total personas con apellido Capacho:"
                }
        ]
) ;
```

Ejemplo 2: Total de estudiantes con edad < 30 años

```
> db.estudiantes.find(
        {
                edad : {  $lt:30}
        }
).count()  ;

> db.estudiantes.aggregate(
        [
                {
                        $match : {
                                edad : { $lt : 30}
                        }
                } ,
                {
                        $count : "Total de Estudiantes edad < 30: "
                }
        ]
)  ;
```

Ejemplo 3: Total de damas con edad >= 50.

```
> db.estudiantes.find(
        {
                "genero":{$eq:"f"} ,
                "edad":{$gte:50}
        }
).count() ;

> db.estudiantes.aggregate(
        [
                {
                        $match : {
                                "genero":{$eq:"f"} ,
                                "edad":{$gte:50}
                        }
                } ,
                {
                        $count: "total damas con edad >= 50: "
                }
        ]
) ;
```

11.3.- {$sum: '$campo_a_sumar'} / {$avg: '$campo_a_promediar'}

Variable: {$sum: '$campo_a_sumar'} --------> Sumatoria
variable: {$avg: '$campo_a_promediar'} ----> Promedio

Ejemplo 1: Obtener la Sumatoria, el Promedio y total de las edades de todos los estudiantes

_id: null ----------→ Indica todos los documentos
suma: {$sum: '$edad'} ----------→ Suma todas las edades
promedio: {$avg:'$edad'} ----------→ Obtiene la edad promedio
total: {$sum: 1} ----------→ Contador de edades

```
> db.estudiantes.aggregate(
    [
        {
            $group : {
                _id: null ,
                suma: { $sum: '$edad' } ,
                promedio: { $avg:'$edad' } ,
                total: { $sum: 1 }
            }
        }
    ]
) ;
```

Ejemplo 2: Obtener la Suma, el Promedio y total de damas con edad >= 50

```
> db.estudiantes.aggregate(
    [
        {
            $match : {
                "genero": {$eq:"f"} ,
                "edad": {$gte:50}
            }
        } ,
        {
            $group : {
                _id: null  ,
                suma: { $sum: '$edad' } ,
                promedio: { $avg:'$edad' } ,
                total: { $sum: 1 }
            }
        }
    ]
) ;
```

📌 Si el campo para la suma no es numérico o no existe: **$sum** lo trata como 0. Esto evita errores y permite que la suma continúe con los valores válidos.

11.4.- {$max: '$campo_a_evaluar'} / {$min: '$campo_a_evaluar'}

Variable: {$max: '$campo_a_evaluar'} ----> Valor Máximo

variable: {$min: '$campo_a_evaluar'} ----> Valor Mínimo

Ejemplo 1: Determinar la edad máxima y mínima en todos los estudiantes

```
> db.estudiantes.aggregate(
    [
        {
            $group : {
                _id: null ,
                edad_max: { $max: '$edad' } ,
                edad_min: { $min:'$edad' } ,
                sumatoria_edad: { $sum: 1 }
            }
        }
    ]
) ;
```

Ejemplo 2: Determinar la fecha de nacimiento mínima y máxima de todos los estudiantes

```
> db.estudiantes.aggregate(
    [
        {
            $group : {
                _id: null ,
                Fecha_minima: { $max: '$fenac' } ,
                Fecha_maxima: { $min:'$fenac' }
            }
        }
    ]
) ;
```

Ejemplo 3: Determinar la edad máxima y mínima de las personas con apellido "Capacho"

```
> db.estudiantes.aggregate(
    [
        {
            $match : {
                apellido :  "Capacho"

            }
        } ,
        {
            $group : {
                _id: null  ,
                edad_max: { $max: '$edad' } ,
                edad_min: { $min:'$edad' } ,
                sumatoria_edad: { $sum: 1 }
            }
        }
    ]
) ;
```

Ejemplo 4: Determinar la edad máxima y mínima de las Damas del semestre 3

```
> db.estudiantes.find(
        { genero:"f"  ,  semestre:3  } ,
        { genero:1,  semestre:1,  edad:1  }
).toArray() ;

> db.estudiantes.aggregate(
        [
                {
                        $match : {  genero:  "f"  ,  semestre:  3  }
                } ,
                {
                        $project: {  // Proyectar (Campos a mostrar)
                                _id: 1,
                                nombre: 1,
                                genero: 1,
                                semestre:1,
                                edad: 1
                        }
                }
        ]
) ;
```

```
> db.estudiantes.aggregate(
    [
        {
            $match : {
                genero: "f" ,
                semestre: 3
            }
        } ,
        {
            $group : {
                _id: null ,
                edad_max: { $max: '$edad' } ,
                edad_min: { $min:'$edad' }
            }
        }
    ]
) ;
```