Guía # 8 – MongoDB

(Agregación/Agrupación/Borrar)

17.- AGRECACIÓN

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

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 : {
                _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' }
            }
        }
    ]
) ;
```

Analizar estas dos salidas

SALIDA # 1

```
> db.estudiantes.aggregate(
    [
        {
            $match : {  genero:  "f"  ,    semestre:  3   }
        } ,
        {
            $project : {
                _id: 1  ,  Nombre: 1  ,  genero:1  ,
                semestre:1  ,  edad:1
            }
        } ,
        {
            $group : {
                _id: null  ,
                edad_max: {  $max: '$edad'  } ,
                edad_min: {  $min:'$edad'  }
            }
        }
    ]
) ;
```

SALIDA #2

```
> db.estudiantes.aggregate(
    [
        {
            $match : {  genero:  "f"  ,  semestre:  3  }
        } ,
        {
            $project : {
                _id: 1  ,  Nombre: 1  ,  genero:1  ,
                semestre:1  ,  edad:1
            }
        } ,
        {
            $group : {
                _id: null  ,
                edad_max: {  $max: '$edad'  } ,
                edad_min: {  $min:'$edad'  }
            }
        }
    ]
) ;
```

17.5.- {$first: '$campo_a_evaluar'} / {$last: '$campo_a_evaluar'}

Variable : { $first: "$campo" } --→ Primer Variable

Variable : { $last: "$campo" } --→ Último Variable

IMPORTANTE con el uso de las funciones $first / $last

* Siempre se usa dentro de un $group
* Para que $first / $last sea realmente útil y predecible: **Se construye primero la etapa $sort y luego la etapa $group.**
* Si no usas $sort - $group con $first / $last; entonces se devolverá un documento arbitrario que elija MongoDB.

Ejemplo 1: Determinar la primera y última edad de todos los estudiantes

```
> db.estudiantes.find(
        { } ,
        { _id:1,  nombre:1,  genero:1 ,  semestre:1,  edad:1  }
).sort( { edad:1 } ).toArray() ;


> db.estudiantes.aggregate(
        [
                {
                        $sort : {  edad:1  }
                } ,
                {
                        $group : {
                                _id: null  ,
                                Primera_edad  : { $first: '$edad' } ,
                                Ultima_edad  : { $last:'$edad' }
                        }
                }
        ]
) ;
```

Ejemplo 2: Mostrar todos los datos de los estudiantes con menor y mayor edad

🚀 **NOTA:** La variable **$$ROOT** hace referencia al documento completo que se está procesando en ese momento en el pipeline - tubería.

```
> db.estudiantes.aggregate(
    [
        {
            $sort : { edad:1 }
        } ,
        {
            $group : {
                _id: null ,
                Primera_edad : { $first: '$edad' } ,
                datos_primera_edad : { $first: "$$ROOT" } ,
                Ultima_edad : { $last:'$edad' } ,
                datos_ultima_edad : { $last: "$$ROOT" }
            }

        }
    ]
) ;
```

Ejemplo 3:

Determinar la primera y última edad de las damas que están estudiando el segundo semestre

```
> db.estudiantes.find(
      { genero : {$eq: "f" }  , semestre : {$eq: 2 } } ,
      { _id:1,  nombre:1, semestre:1,  genero:1 ,  edad:1 }
).sort( { edad:1 } ).toArray() ;

> db.estudiantes.aggregate(
      [
            {
                  $match : {
                        genero : {$eq: "f" }  ,
                        semestre  : {$eq: 2 }
                  }
            } ,

            {
                  $sort : { edad:1 }
            } ,
```

```
        {
            $group : {
                _id: null  ,
                Primera_edad : { $first: '$edad' } ,
                Ultima_edad  : { $last:'$edad' }
            }
        }
    ]
) ;
```

🚀 **NOTA:** La variable **$$ROOT** hace referencia al documento completo que se está procesando en ese momento en el pipeline - tubería.

```
> db.estudiantes.aggregate(
    [
        {
            $match : {
                genero : {$eq: "f" }  ,
                semestre  : {$eq: 2 }
            }
        } ,
```

```
218
219             {
220                 $sort : {  edad:1  }
221             } ,
222             {
223                 $group : {
224                     _id: null  ,
225                     Primera_edad  : { $first: '$edad' } ,
226                     datos_primera_edad  : { $first: "$$ROOT" } ,
227                     Ultima_edad  : { $last:'$edad' } ,
228                     datos_ultima_edad  : { $last: "$$ROOT" }
229                 }
230
231             }
232         ]
233 ) ;
234
235
236
237
238
239
240
241
```

17.6.- Agrupación: Es posible agrupar varios datos en un **_id**.
MongoDB agrupa los documentos basándose en la combinación única de los valores de esos campos.

📝 Sintaxis
Para agrupar por múltiples campos, defines el _id como un objeto y dentro de él especificas los campos que deseas usar para la agrupación:

```
{
    $group: {
        // El _id ahora es un objeto que combina 'campo1' y 'campo2'
        _id: {
            campo1: '$nombreDelCampo1' ,
            campo2: '$nombreDelCampo2'
        } ,
        // Otras funciones de agregación
        conteo: { $sum: 1 }
    }
}
```

Ejemplo 1: Generar un Listado del Total de estudiantes por cada edad

```
> db.estudiantes.aggregate(
    [
        {
            $group : {
                _id: "$edad"    ,
                Total : { $sum: 1 }   ,
            }

        }
    ]
) .toArray() ;

> db.estudiantes.aggregate(
    [
        {
$group : { _id: "$edad"  , Total : { $sum: 1 } }
        } ,
        {    $sort : { _id: 1 }    }
    ]
) .toArray() ;
```

Ejemplo 2: Generar un listado del total de estudiantes damas con edad >= 30
Se debe mostrar: Edad, la sumatoria de la edad, el promedio de la edad y el total de edades

```
> db.estudiantes.aggregate(
    [
        {
            $match: {
                genero: "f" , edad: { $gte:30 }
            }
        } ,
        {
            $group : {
                _id: "$edad" ,
                suma : { $sum: "$edad" } ,
                promedio : { $avg: "$edad" } ,
                total : { $sum: 1 }
            }
        }
    ]
) .toArray() ;
```

Datos ordenados edad

```
> db.estudiantes.aggregate(
    [
        {
            $match: {
                genero: "f" , edad: { $gte:30 }
            }
        } ,
        {
            $group : {
                _id: "$edad" ,
                suma : { $sum: "$edad" } ,
                promedio : { $avg: "$edad" } ,
                total : { $sum: 1 }
            }

        } ,
        { $sort : { _id: 1 } }
    ]
) .toArray() ;
```

Ejemplo 3: Mostrar un reporte que agrupe los datos por semestre y edad debidamente ordenados en forma ascendente

```
> db.estudiantes.aggregate(
    [
        {
            $match:{  semestre  :  {$gte:1}  }
        } ,
        {
            $group : {
                _id:{
                    semestre  :  "$semestre"  ,
                    edad      :  "$edad"

                } ,
                total  :  {  $sum: 1  }   ,
            }

        }  ,
        {  $sort : {  "_id.semestre":  1, "_id.edad": 1  }  }
    ]
) .toArray() ;
```

18.- BORRAR UNO O VARIOS CAMPOS DE UNA COLECCIÓN 🗑️

Para borrar (eliminar) un campo específico de un documento en MongoDB, se debe usar el método **updateOne** o **updateMany** junto con el operador **$unset**.

El operador $unset ----> elimina un campo del documento.

🛠️ Sintaxis Básica:

```
> db.coleccion.updateOne(
      { <criterio_de_busqueda> } ,          // Documento que deseas modificar
      { $unset: { <nombre_del_campo>: ""/"1" }   } // Operación: $unset
) ;
```

Ejemplo 1: Se desea borrar el campo programa y el campo fenac del estudiante cuyo id es 44

```
> db.estudiantes.find(   {  _id: {$eq:44}    }   )   ;
[
      {
            _id: 44,
            nombre: 'Daniel',
            apellido: 'Zambrano',
            genero: 'm',
            estado: 'Casado',
            fenac: '1987-01-01',
            edad: 37,
            programa: [ 'C', 'Pascal', 'Fortran', 'Redes', 'Cobol' ],
            semestre: 4
      }
]
```

```
> db.estudiantes.updateOne(
        {   _id: { $eq: 44 } } ,
        { $unset: {   programa: "" }   }
) ;

> db.estudiantes.updateOne(
        {   _id: { $eq: 44 } } ,
        { $unset: {   fenac: "1" }   }
) ;

> db.estudiantes.find( {  _id: {$eq:44} } );
[
        {
                _id: 44,
                nombre: 'Daniel',
                apellido: 'Zambrano',
                genero: 'm',
                estado: 'Casado',
                edad: 37,
                semestre: 4
        }
]
```

Nota: Si el objeto de filtro está vacío (**{}**), se aplica a todos los documentos de la colección.

🗑️ **Borrar Múltiples Campos en un solo documento**

Ejemplo 2: Se desea borrar los campos: estado, edad y materias del estudiante cuyo id es 57

```
> db.estudiantes.find(  { apellido: {$eq:  "Capacho"}  }  )  ;
[
        {
                _id: 57,
                nombre: 'Sergio',                apellido: 'Capacho',
                genero: 'm',                     estado: 'Unión Libre',
                fenac: '1967-11-22',             edad: 55,
                materias: {
                        programacion1: 5,
                        programacion2: 5,
                        programacion3: 5,
                        bd1: 5,
                        bd2: 5
                },
                semestre: 7
        }
]

> db.estudiantes.updateOne(
        {  _id: { $eq: 57 } } ,
        {
                $unset: {
                        estado: "1"  ,  edad:  "",  materias: "1"
                }
        }
) ;
```

19.- BORRAR UNA COLECCIÓN 🗑

> use MyDataBase

MyDataBase> db.**nombre_de_la_coleccion.**drop() ;

Ejemplo 1:

Test> Use miUniversidad

miUniversidad> show Collections ;

estudiantes

materias

miUniversidad> db.**materias.**drop() ;

miUniversidad> show Collections ;

estudiantes

20.- BORRAR UNA BASE DE DATOS 🗑

colegio01>use myDataBase

myDataBase>db.**dropDatabase()** ;

Ejemplo 1:

Test> Use miUniversidad

miUniversidad> db.**dropDataBase()** ;