

Guía # 8 – MongoDB

(Agregación/Agrupación/Borrar)

17.- AGREGACIÓN

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  
            }  
        }  
    ]  
);
```

24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

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

49

50 Analizar estas dos salidas

51 SALIDA # 1

52 > db.estudiantes.aggregate(
53

53 [

54 {

55 \$match : { genero: "f" , semestre: 3 }

56 } ,

57 {

58 \$project : {

59 _id: 1 , Nombre: 1 , genero:1 ,

60 semestre:1 , edad:1

61 }

62 } ,

63 {

64 \$group : {

65 _id: null ,

66 edad_max: { \$max: '\$edad' } ,

67 edad_min: { \$min: '\$edad' }

68 }

69 }

70]

71) ;

72

73 SALIDA #2

74 > db.estudiantes.aggregate(
75

[

76

{

77

\$match : { genero: "f" , semestre: 3 }

78

},

79

{

80

\$project : {

81

_id: 1 , Nombre: 1 , genero:1 ,

82

semestre:1 , edad:1

83

}

84

},

85

{

86

\$group : {

87

_id: null ,

88

edad_max: { **\$max**: '\$edad' } ,

89

edad_min: { **\$min**: '\$edad' }

90

}

91

}

92

]

93

);

94

95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125

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.

```


126
127 Ejemplo 1: Determinar la primera y última edad de todos los estudiantes
128 > db.estudiantes.find(
129     { } ,
130     { _id:1, nombre:1, genero:1, semestre:1, edad:1 }
131 ).sort( { edad:1 } ).toArray() ;
132
133
134 > db.estudiantes.aggregate(
135     [
136         {
137             $sort : { edad:1 }
138         } ,
139         {
140             $group : {
141                 _id: null ,
142                 Primera_edad : { $first: '$edad' } ,
143                 Ultima_edad : { $last: '$edad' }
144             }
145         }
146     ]
147 ) ;
148

```

149

150

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

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

154 > db.estudiantes.**aggregate(**

155 **[**

156 **{**

157 **\$sort : { edad:1 } ,**

158 **}** ,

159 **{**

160 **\$group : {**

161 _id: null ,

162 Primera_edad : { **\$first: '\$edad'** } ,

163 datos_primera_edad : { **\$first: "\$\$ROOT"** } ,

164 Ultima_edad : { **\$last: '\$edad'** } ,

165 datos_ultima_edad : { **\$last: "\$\$ROOT"** }

166 **}**

167

168 **}**

169 **]**

170 **) ;**

171

172

173

174

175 **Ejemplo 3:**

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

178 > db.estudiantes.find(
179

179 { genero : {\$eq: "f" } , semestre : {\$eq: 2 } } ,

180 { _id:1, nombre:1, semestre:1, genero:1, edad:1 }

181).sort({ edad:1 }).toArray() ;

182

183 > db.estudiantes.aggregate(
184

184 [

185 {

186 \$match : {

187 genero : {\$eq: "f" } ,

188 semestre : {\$eq: 2 }

189 }

190 } ,

191

192 {

193 \$sort : { edad:1 }

194 } ,


195

```

196
197     {
198         $group : {
199             _id: null ,
200             Primera_edad : { $first: '$edad' } ,
201             Ultima_edad : { $last: '$edad' }
202         }
203     }
204 ]
205 ) ;

```

206
207

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

```

210 > db.estudiantes.aggregate(
211     [
212         {
213             $match : {
214                 genero : {$eq: "f" } ,
215                 semestre : {$eq: 2 }
216             }
217         } ,

```

```
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
```

242

243 **17.6.- Agrupación:** Es posible agrupar varios datos en un `_id`.

244 MongoDB agrupa los documentos basándose en la combinación única de los valores de
245 esos campos.

246

247  Sintaxis

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

250

251 {

252 **`$group: {`**

253 `// El _id ahora es un objeto que combina 'campo1' y 'campo2'`

254 **`_id: {`**

255 **`campo1: '$nombreDelCampo1' ,`**

256 **`campo2: '$nombreDelCampo2'`**

257 **`} ,`**

258 `// Otras funciones de agregación`

259 `conteo: { $sum: 1 }`

260 **`}`**

261 **`}`**

262

263

264

265

266

267

268

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

270

271 > db.estudiantes.**aggregate**(

272 [

273 {

274 **\$group** : {

275 **_id**: "\$edad" ,

276 Total : { **\$sum**: 1 } ,

277 }

278

279 }

280]

281 **) .toArray() ;**

282

283 > db.estudiantes.**aggregate**(

284 [

285 {

286 **\$group** : { **_id**: "\$edad" , Total : { **\$sum**: 1 } }

287 } ,

288 { **\$sort** : { **_id**: 1 } }

289]

290 **) .toArray() ;**

291

292 **Ejemplo 2:** Generar un listado del total de estudiantes damas con edad >= 30

293 Se debe mostrar: Edad, la sumatoria de la edad, el promedio de la edad y
294 el total de edades

295

296 > db.estudiantes.**aggregate(**

297

[

298

{

299

\$match: {

300

genero: "f" , edad: { \$gte:30 }

301

}

302

},

303

{

304

\$group : {

305

_id: "\$edad" ,

306

suma : { \$sum: "\$edad" } ,

307

promedio : { \$avg: "\$edad" } ,

308

total : { \$sum: 1 }

309

}

310

}

311

]

312

313 **) .toArray() ;**

314

```

315
316 Datos ordenados edad
317 > db.estudiantes.aggregate(
318     [
319         {
320             $match: {
321                 genero: "f" , edad: { $gte:30 }
322             }
323         } ,
324         {
325             $group : {
326                 _id: "$edad" ,
327                 suma : { $sum: "$edad" } ,
328                 promedio : { $avg: "$edad" } ,
329                 total : { $sum: 1 }
330             }
331         } ,
332         { $sort : { _id: 1 } }
333     ]
334 ) .toArray() ;
335
336
337
338

```

339

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

```
342 > db.estudiantes.aggregate(  
343     [  
344         {  
345             $match: { semestre : {$gte:1} }  
346         } ,  
347         {  
348             $group : {  
349                 _id: {  
350                     semestre : "$semestre" ,  
351                     edad      : "$edad"  
352                 } ,  
353                 total : { $sum: 1 } ,  
354             }  
355         }  
356     ] ,  
357     { $sort : { "_id.semestre": 1, "_id.edad": 1 } }  
358 ]  
359 ) .toArray() ;  
360  
361  
362
```

363

364 **18.- BORRAR UNO O VARIOS CAMPOS DE UNA COLECCIÓN**

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

367

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

369

370  Sintaxis Básica:

371 **> db.coleccion.updateOne(**

372 **{ <critero_de_búsqueda> } ,** // Documento que deseas modificar

373 **{ \$unset: { <nombre_del_campo>: ""/"1" } }** // Operación: \$unset

374 **) ;**

375

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

378 **> db.estudiantes.find({ _id: {\$eq:44} }) ;**

379 [
380 {

381 _id: 44,

382 nombre: 'Daniel',

383 apellido: 'Zambrano',

384 genero: 'm',

385 estado: 'Casado',

386 fenac: '1987-01-01',

387 edad: 37,

388 programa: ['C', 'Pascal', 'Fortran', 'Redes', 'Cobol'],

389 semestre: 4

390 }

391]

```
392
393
394 > db.estudiantes.updateOne(
395     { _id: { $eq: 44 } } ,
396     { $unset: { programa: "" } }
397 ) ;
398
399 > db.estudiantes.updateOne(
400     { _id: { $eq: 44 } } ,
401     { $unset: { fenac: "1" } }
402 ) ;
403
404 > db.estudiantes.find( { _id: { $eq: 44 } } );
405 [
406     {
407         _id: 44,
408         nombre: 'Daniel',
409         apellido: 'Zambrano',
410         genero: 'm',
411         estado: 'Casado',
412         edad: 37,
413         semestre: 4
414     }
415 ]
416
417
418 Nota: Si el objeto de filtro está vacío ( {} ), se aplica a todos los documentos de la
419 colección.
420
```

421

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

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

```
425 > db.estudiantes.find( { apellido: {$eq: "Capacho"} } ) ;
426 [
427   {
428     _id: 57,
429     nombre: 'Sergio',           apellido: 'Capacho',
430     genero: 'm',               estado: 'Unión Libre',
431     fenac: '1967-11-22',      edad: 55,
432     materias: {
433       programacion1: 5,
434       programacion2: 5,
435       programacion3: 5,
436       bd1: 5,
437       bd2: 5
438     },
439     semestre: 7
440   }
441 ]
442
443 > db.estudiantes.updateOne(
444   { _id: { $eq: 57 } } ,
445   {
446     $unset: {
447       estado: "1" ,   edad: "",   materias: "1"
448     }
449   }
450 ) ;
```

451

452 **19.- BORRAR UNA COLECCIÓN**

453

454 **> use MyDataBase**

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

456 **Ejemplo 1:**

457 Test> Use miUniversidad

458 miUniversidad> show Collections ;

459 estudiantes

460 materias

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

462 miUniversidad> show Collections ;

463 estudiantes

464

465 **20.- BORRAR UNA BASE DE DATOS**

466 colegio01>use myDataBase

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

468

469 **Ejemplo 1:**

470 Test> Use miUniversidad

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

472