

# 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} }  
        ]  
      }  
    ]  
  }  
) ;
```

23

24

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

27 **A.- Usando \$in**

```
28 > db.estudiantes.countDocuments(  
29     {  
30         programa:{  
31             $in: ["Python", "Cobol", "java"]  
32         }  
33     }  
34 ) ;
```

35

36 **B.- Usando \$or**

```
37 > db.estudiantes.countDocuments(  
38     {  
39         $or:[  
40             { programa: "Python" } ,  
41             { programa: "Cobol" } ,  
42             { programa: "Java" }  
43         ]  
44     }  
45 ) ;
```

46

47

## 48 **17.- AGREGACIÓN**

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

52

53 La agregación es una forma de procesar un gran número de documentos de una  
54 colección haciéndolos pasar por distintas etapas.

55 Donde cada etapa constituye lo que se conoce como "pipeline (**tuberías**)", y consiste en  
56 un paso donde se filtran, ordenan, agrupan, remodelan o modifican los documentos que  
57 pasan por el pipeline (**tubo**).

58

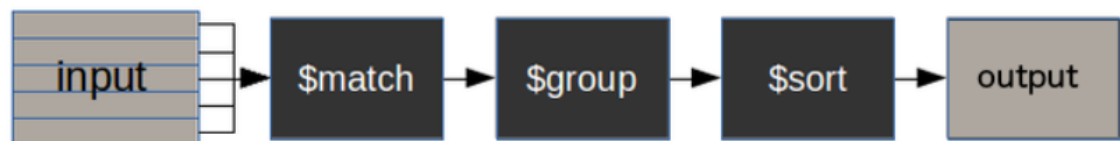
59

60

61

62

63



64

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

65

```
db.collection.aggregate(
```

66

```
[
```

67

```
  { <etapa1 > } ,
```

68

```
  { <etapa2 > } ,
```

69

```
  *
```

70

```
  *
```

71

```
  { <etapaN > }
```

72

```
]
```

73

```
);
```

74

75

76 **17.1.- \$match: { } ---> Igual / find()**

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

80

81 Se pueden usar los operadores:

82 • Relacionales:

83 ○ Igualdad: { campo: valor } / { campo: {\$eq: valor}}

84 ○ Comparación:

85 ▪ \$gt (mayor que) / \$gte(mayor o igual que),

86 ▪ \$lt (menor que) / \$lte (menor o igual que),

87 ▪ \$ne (no igual).

88 • Lógicos: \$and, \$or, \$not.

89 • Elementos de Array: \$in, \$nin.

90 • Otros: \$exists, \$regex, etc..

91 > db.estudiantes.**aggregate()**

92 [

93 {

94 **\$match: {**

95 <pregunta1> ,

96 <pregunta2> ,

97 \* ,

98 <preguntaN>

99 **}**

100 }

101 ]

102 **);**

Sintaxis

103

104 **Ejemplo 1:** Extraer todos los documentos donde este el nombre Zila

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

106

107

108 > db.estudiantes.aggregate(

109 [

110 {

111 **\$match :** {

112 "nombre" : "Zila"

113 **}**

114 }

115 ]

116 **) ;**

117

118

119 > db.estudiantes.aggregate(

120 [

121 {

122 **\$match :** {

123 "nombre" : { \$eq: "Zila" }

124 **}**

125 }

126 ]

127 **) ;**

128

```

129
130 Ejemplo 2 ($AND) : Mostrar las Damas con Edad >= 50
131 > db.estudiantes.find(
132     {
133         $and:[
134             {"genero":{"$eq":"f"}} } ,
135             {"edad":{"$gte:50"}} }
136         ]
137     }
138 ).toArray() ;
139
140
141 > db.estudiantes.find(
142     {
143         "genero":{"$eq":"f"},
144         "edad":{"$gte:50}
145     } ,
146     {
147         _id:1 ,
148         nombre: 1 ,
149         genero: 1 ,
150         edad: 1
151     }
152 ) ;
153
154

```

155

156 > db.estudiantes.aggregate(

157 [

158 {

159 **\$match** : {

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

161 "edad":{\$gte:50}

162 }

163 } ,

164 {

165 **\$project**: { // Proyectar (Campos a mostrar)

166 \_id: 1,

167 nombre: 1,

168 genero: 1,

169 edad: 1

170 }

171 }

172 ]

173 ) ;

174

175

176

177

178

179

180

181

182

183

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

185

186 > db.estudiantes.**aggregate**(

187

[

188

{

189

**\$match : {**

190

**\$or:[**

191

{ edad : {\$eq:24} } ,

192

{ edad : {\$eq:37} }

193

]

194

195

**}**

196

**},**

197

{

198

**\$project: {** // Proyectar (Campos a mostrar)

199

\_id: 1,

200

nombre: 1,

201

genero: 1,

202

edad: 1

203

**}**

204

}

205

]

206

**) ;**

207

208

209

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

211

212 > db.estudiantes.**aggregate(**

213     **[**

214         **{**

215             **\$match : {**

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

217             **\$or:[**

218                 { edad : {\$eq:24} } ,

219                 { edad : {\$eq:37} } ,

220             **]**

221

222         **}**

223     **},**

224     **{**

225         **\$project: {** // Proyectar (Campos a mostrar)

226             \_id: 1,

227             nombre: 1,

228             genero: 1,

229             edad: 1

230         **}**

231     **}**

232     **]**

233 **) ;**

234

235

236

237 **17.2.- { \$count: } ---> Contar**

238

239 **Ejemplo 1:** Total de estudiantes con el apellido Capacho

240 > db.estudiantes.find(  
241     {  
242         Apellido: 'Capacho'  
243     }  
244 ).count() ;

245

246 > db.estudiantes.**aggregate**(

247     [  
248         {  
249             **\$match** : {  
250                 apellido : "Capacho"  
251             }  
252         },  
253         {  
254             **\$count**:"Total personas con apellido Capacho:"  
255         }  
256     ]  
257     )  
258     ;

251

259

260

261

262

263

264

265 **Ejemplo 2:** Total de estudiantes con edad < 30 años

266 > db.estudiantes.find(  
267     {  
268         edad : { \$lt:30}  
269     }  
270 ).count() ;  
271

271

272 > db.estudiantes.**aggregate**(

273     [  
274         {  
275             **\$match** : {  
276                 edad : { \$lt : 30}  
277             }  
278         } ,  
279         {  
280             **\$count** : "Total de Estudiantes edad < 30: "  
281         }  
282     ]  
283 **) ;**  
284  
285  
286  
287  
288

283

284

285

286

287

288

```

289
290
291
292 Ejemplo 3: Total de damas con edad >= 50.
293 > db.estudiantes.find(
294     {
295         "genero":{"$eq:"f"} ,
296         "edad":{"$gte:50}
297     }
298 ).count() ;
299
300 > db.estudiantes.aggregate(
301     [
302         {
303             $match : {
304                 "genero":{"$eq:"f"} ,
305                 "edad":{"$gte:50}
306             }
307         } ,
308         {
309             $count: "total damas con edad >= 50: "
310         }
311     ]
312 ) ;
313
314

```

315

316 **11.3.-** `{ $sum: '$campo_a_sumar' } / { $avg: '$campo_a_promediar' }`

317 Variable: `{ $sum: '$campo_a_sumar' }` -----> Sumatoria

318 variable: `{ $avg: '$campo_a_promediar' }` ----> Promedio

319

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

322

323 `_id: null` -----> Indica todos los documentos

324 `suma: { $sum: '$edad' }` -----> Suma todas las edades

325 `promedio: { $avg: '$edad' }` -----> Obtiene la edad promedio

326 `total: { $sum: 1 }` -----> Contador de edades

327

328 `> db.estudiantes.aggregate(`

329 `[`

330 `{`

331 `$group : {`

332 `_id: null ,`

333 `suma: { $sum: '$edad' } ,`

334 `promedio: { $avg: '$edad' } ,`

335 `total: { $sum: 1 }`

336 `}`

337 `}`

338 `]`

339 `);`

340

341

342

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

344

345 > db.estudiantes.**aggregate**(

346

[

347

{

348

**\$match** : {

349

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

350

"edad": {\$gte:50}

351

}

352

},

353

{

354

**\$group** : {

355

\_id: null ,

356

suma: { **\$sum**: '\$edad' } ,

357

promedio: { **\$avg**: '\$edad' } ,

358

total: { **\$sum**: 1 }

359

}

360

}

361

]

362


) ;

363

364

365

366

367  Si el campo para la suma no es numérico o no existe: **\$sum** lo trata como 0.

368 Esto evita errores y permite que la suma continúe con los valores válidos.

369

370

371 **11.4.- {\$max: '\$campo\_a\_evaluar'} / {\$min: '\$campo\_a\_evaluar'}**

372 Variable: {\$max: '\$campo\_a\_evaluar'} ----> Valor Máximo

373 variable: {\$min: '\$campo\_a\_evaluar'} ----> Valor Mínimo

374

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

376

377 > db.estudiantes.**aggregate(**

378 [

379 {

380 **\$group : {**

381

\_id: null ,

382

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

383

edad\_min: { **\$min:'\$edad'** } ,

384

sumatoria\_edad: { **\$sum: 1** }

385

**}**

386

**}**

387

**]**

388

**) ;**

389

390

391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417

**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' }  
      }  
    }  
  ]  
);
```

418

419

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

422

423 > db.estudiantes.**aggregate**(

424

[

425

{

426

**\$match** : {

427

apellido : "Capacho"

428

429

}

430

},

431

{

432

**\$group** : {

433

\_id: null ,

434

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

435

edad\_min: { **\$min**: '\$edad' } ,

436

sumatoria\_edad: { **\$sum**: 1 } ,

437

}

438

}

439

]

440

) ;

441

442

443

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

445 > db.estudiantes.find(  
446

446 { genero:"f" , semestre:3 } ,

447 { genero:1, semestre:1, edad:1 }  
448

448 ).toArray() ;  
449

449

450 > db.estudiantes.**aggregate**(

451 [

452 {

453 **\$match** : { genero: "f" , semestre: 3 }  
454

454 } ,

455 {

456 **\$project**: { // Proyectar (Campos a mostrar)

457 \_id: 1,

458 nombre: 1,

459 genero: 1,

460 semestre:1,

461 edad: 1  
462

462 }  
463

463 }  
464

464 ]  
465

465 ) ;  
466

466

467

468

469

470 > db.estudiantes.aggregate(**(**

471 [

472 {

473 **\$match** : {

474 genero: "f" ,

475 semestre: 3

476 }

477 } ,

478 {

479 **\$group** : {

480 \_id: null ,

481 edad\_max: { **\$max**: '\$edad' } ,

482 edad\_min: { **\$min**: '\$edad' }

483 }

484 }

485 ]

486 ) ;

487