

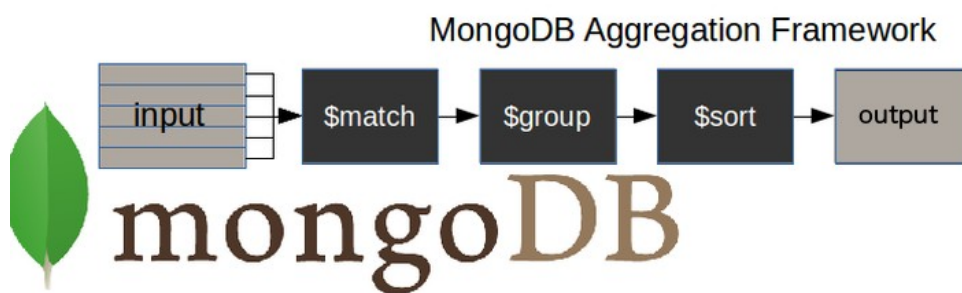
## CONSULTAS Y TUBERÍAS DE AGREGACIÓN

La **agregación** es una forma de procesar un gran número de documentos de una colección haciéndolos pasar por distintas etapas. Las etapas constituyen lo que se conoce como "**pipeline**". Las etapas de un pipeline pueden filtrar, ordenar, agrupar, remodelar y modificar los documentos que pasan por el pipeline.

Uno de los usos más comunes de la agregación es calcular valores agregados para grupos de documentos. Esto es similar a la agregación básica disponible en SQL con la cláusula GROUP BY y las funciones COUNT, SUM y AVG. Sin embargo, MongoDB Aggregation va más allá y también puede realizar uniones de tipo relacional, remodelar documentos, crear nuevas colecciones y actualizar las existentes, etc.

Aunque existen otros métodos para obtener datos agregados en MongoDB, el marco de agregación es el enfoque recomendado para la mayoría de los trabajos.

El proceso de agregación funciona de la siguiente manera:



- ***\$match*** - filtra los documentos con los que necesitamos trabajar, los que se ajustan a nuestras necesidades
- ***\$group*** - realiza el trabajo de agregación, realizando consultas de agregación o resumen, como totales, medias, máximos, ... Entre los operadores de agregación podemos citar \$count, \$max, \$min, \$avg, \$suma
- ***\$sort*** - ordena los documentos resultantes de la forma que deseemos (ascendente o descendente)

La entrada de la cadena puede ser una sola colección, en la que se pueden fusionar otras más adelante.

A continuación, la tubería realiza transformaciones sucesivas en los datos hasta alcanzar nuestro objetivo.

De este modo, podemos dividir una consulta compleja en etapas más sencillas, en cada una de las cuales completamos una operación diferente sobre los datos. Así, al final de la cadena de consultas, habremos conseguido todo lo que queríamos.

Este enfoque nos permite comprobar si nuestra consulta funciona correctamente en cada etapa examinando tanto su entrada como su salida. La salida de cada etapa será la entrada del siguiente.

La sintaxis es la siguiente:

```
db.collectionName.aggregate(pipeline, options)
```

donde

- *collectionName* - es el nombre de una colección
- *pipeline* - es un array que contiene las etapas de agregación
- *options* - parámetros opcionales para la agregación

Este es un ejemplo de la sintaxis del canal de agregación:

```
pipeline = [  
    { $match : { ... } },  
    { $group : { ... } },  
    { $sort : { ... } }  
]
```

- ***\$project*** – recupera solo los campos que nos interesan
- ***\$unwind*** – para trabajar con los valores de los campos dentro de un array.

No se puede trabajar directamente sobre los elementos de un array dentro de un documento con etapas como \$group. *\$unwind* nos permite trabajar con los valores de los campos dentro de un array.

Si en los documentos de entrada hay un campo array , a veces tendrá que imprimir el documento varias veces, una por cada elemento de array.

En cada copia del documento se sustituye el campo array por el elemento sucesivo.

- ***\$limit*** – para limitar el número de documentos ordenados
- ***\$lookup*** – para cuando necesitamos utilizar información de más de una colección. Pero debemos tener en cuenta que, dado que MongoDB se basa en documentos, podemos darles la forma que necesitemos.

Vamos a trabajar con el entorno MongoDB Atlas y con sus bases de datos de ejemplo.

1)

Base de datos: sample\_airbnb

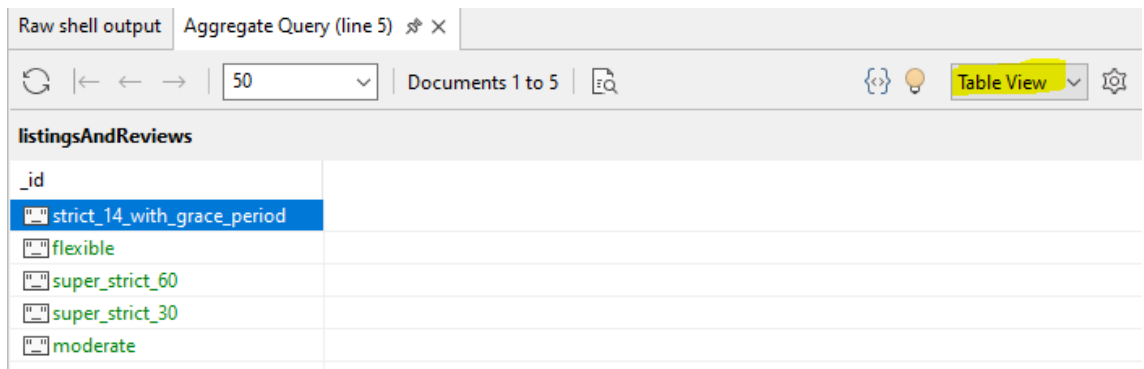
Colección: listingsAndReviews

Encontrar las distintas políticas de cancelación.

```
use sample_airbnb
```

```
already on db sample_airbnb
```

```
db.listingsAndReviews.aggregate([
  {$group: { // Agrupa documentos por el campo cancellation_policy
    _id: "$cancellation_policy"}}
])
```



listingsAndReviews	
_id	
strict_14_with_grace_period	
flexible	
super_strict_60	
super_strict_30	
moderate	

2)

Base de datos: sample\_airbnb

Colección: listingsAndReviews

Encontrar alojamientos con 4 camas, Wifi y cuyo precio esté entre 80€ y 100€.

```
db.listingsAndReviews.aggregate([
  {$match: {$and: [{"beds": 4},
    {"amenities": "Wifi"},
    {"price": { $gte: 80, $lte: 100 }}}}},
  {$project: {_id: 0, name: 1, beds: 1, amenities: 1, price: 1}}
])
```

```
// Con find
db.listingsAndReviews.find({
  "beds": 4,
  "amenities": "Wifi",
  "price": { $gte: 80, $lte: 100 }
},{name: 1, beds: 1, amenities: 1, price: 1, _id: 0})
```

listingsAndReviews > beds			
name	beds	amenities	price
Next to Barcelo...	4	[ 17 elements ]	90.0
Penthouse. Terr...	4	[ 40 elements ]	87.0
São João	4	[ 15 elements ]	80.0
Apartamento fa...	4	[ 58 elements ]	100.0
Glorias:Sagrada ...	4	[ 43 elements ]	85.0
Countrv Oasis P...	4	[ 20 elements ]	99.0

3)

Base de datos: sample\_airbnb

Colección: listingsAndReviews

Obtener un listado de tipos de alojamiento (room\_type) y un recuento del total de alojamientos de cada tipo.

```
db.listingsAndReviews.aggregate([
  {$group: {_id: "$room_type", total: { $sum: 1 }}}
])
```

listingsAndReviews > total	
_id	total
Entire home/apt	3489.0
Private room	1983.0
Shared room	83.0

4)

Base de datos: sample\_airbnb

Colección: listingsAndReviews

Encontrar alojamientos en "United States" cuya calificación en limpieza (review\_scores\_cleanliness) sea 8 o superior.

```
db.listingsAndReviews.find({
  "address.country": "United States",
  "review_scores.review_scores_cleanliness": { $gte: 8 }
})
```

listingsAndReviews > name					
_id	name	listing_url	space	description	neighborhood_o
1001265	Ocean View W...	https://www.ai...	Great studio lo...	A short distanc...	You can bre
10021707	Private Room i...	https://www.ai...		Here exists a v...	
1003530	New York City ...	https://www.ai...	Murphy bed, o...	Murphy bed, o...	Great neigh
10057826	Deluxe Loft Suite	https://www.ai...	This loft unit fe...	Loft Suite Delu...	Greenpoint i
10096773	Easy 1 Bedroo	https://www.ai...	*I listed this pla	A comfortable	Chelsea is cr

5)

Base de datos: sample\_airbnb

Colección: listingsAndReviews

Obtener un listado de los 10 alojamientos con mayor número de comentarios (reviews).

```
db.listingsAndReviews.aggregate([
  {$match: { // Filtra los documentos que tengan al menos un review
    "reviews": {$exists: true, $ne: []}}},
  {$project: { // Proyecta solo los campos que necesitamos para el conteo
    "name": 1, "reviews_count": {$size: "$reviews"}}},
  {$sort: { // Ordena en orden descendente por el número de comentarios
    "reviews_count": -1}},
  {$limit: 10 // Limita los resultados a 10
  }
])

db.listingsAndReviews.aggregate([
  {$match: {number_of_reviews: {$gte: 1}}}, // Que tengan al menos 1 review
  {$project: {name: 1, number_of_reviews: 1}}, // Indica el proyect para verificarlo
  {$sort: {number_of_reviews: -1}}, // Ordena descendientemente por el número de comentarios
  {$limit: 10} // limita los resultados a 10
])
```

listingsAndReviews > _id		
_id	name	reviews_count
4069429	#Private Studio...	533
12954762	Near Airport pr...	469
95560	La Sagrada Fa...	463
476983	PRIVATE Room ...	420
5283892	traditional and ...	408
2758817	Porto city cent...	402
1284759	ABEL'S IN DO...	399
1482060	Beautiful apart...	397
127208	B & B Room Y...	391
11610598	The Ohana at V...	385

6)

Base de datos: sample\_analytics

Colección: customers

Obtener el nombre (name), el email, el código de la cuenta y su límite, del cliente cuyo username es "serranobrian".

```
use sample_analytics

db.customers.aggregate([
  {$lookup:{
    from: "accounts", // Coleccion con la que se relaciona customers
    localField: "accounts", // Campo local de customers
    foreignField: "account_id", // Campo foraneo con el que se relaciona de accounts
    as: "resultado"}, // Es donde creará el array resultado de la combinación de las 2 relaciones
  {$unwind: "$resultado"}, // Hace el unwind del array resultado
  {$match: {username: "serranobrian"}},
  {$project: {_id: 0, name: 1, username: 1, email: 1, "codigodelacuenta": "$resultado.account_id", "limitedelacuenta": "$resultado.limit"}}
  // Sacamos también el username aunque no lo pide para verificar que es serranobrian
])
```

customers > name				
username	name	email	codigodelacuenta	limitedelacuenta
serranobrian	Leslie Martinez	tcrawford@gm...	170945	10000
serranobrian	Leslie Martinez	tcrawford@gm...	951849	10000

7)

Base de datos: sample\_analytics

Colección: customers

Obtener un listado ascendente del número de clientes por edad.

```
db.customers.aggregate([
  {$group: {
    _id: {$subtract: [{year: (new Date())}, {$year: ("$birthdate")}]}, // Restamos la fecha de hoy con la del cumpleaños
    total: {$sum: 1}}},
  {$sort: {_id: 1}}
])
```

customers > total	
_id	total
27	4.0
28	17.0
29	17.0
30	19.0
31	11.0
32	22.0
33	19.0
34	20.0

8)

Base de datos: sample\_analytics

Colección: transactions

Importe total de las compras (transaction\_code: "buy") realizadas por la cuenta (account\_id) 443178.

```
db.transactions.aggregate([
  {$unwind: "$transactions"}, // Hace el unwind para filtrar dentro de las transacciones embebidas
  {$match: {$and: [{account_id: 443178}, {"transactions.transaction_code": "buy"}]}},
  {$group: {
    _id: "$transactions.transaction_code",
    numerodetransacciones: {$sum: 1},
    importetotal: {$sum: "$transactions.amount"}}
  }
])
```

transactions > numerodetransacciones		
_id	numerodetransacio...	importetotal
buy	45.0	235232

9)

Base de datos: sample\_guides





Colección: planets

Encontrar qué planetas tienen anillos (hasRings). Sólo nos interesa el nombre del planeta y los queremos ordenar según su posición en relación con el sol (orderFromSun).

use sample\_guides

```
db.planets.aggregate([
  {$match: {hasRings: true}},
  {$sort: {orderFromSun: 1}},
  {$project: {_id: 0, name: 1}}
])

// Con find
db.planets.find({"hasRings": true}, {"name": 1, "_id": 0}).sort({"orderFromSun": 1})
```

planets	
name	
 Jupiter	
 Saturn	
 Uranus	
 Neptune	





10)

Base de datos: sample\_guides

Colección: planets

Encontrar los planetas en cuya atmósfera hay Helio (He).

```
db.planets.aggregate([
  {$match:{mainAtmosphere: "He"}},
  {$project:{_id: 0, name: 1}}
])
```

planets	
name	
 Uranus	
 Neptune	
 Jupiter	
 Saturn	

11)

Base de datos: sample\_mflix

Colección: movies

Encontrar películas de aventuras con 10 nominaciones o más. Mostrar sólo el nombre de la película, el año, las nominaciones y la productora.

```
use sample_mflix
```

```
db.movies.aggregate([
  {$match:{$and: [{genres: "Adventure"}, {"awards.nominations": {$gte: 10}}]}},
  {$project: {_id: 0, title: 1, year: 1, "nominaciones": "$awards.nominations", "productora": "$tomatoes.production"}}
])
```



movies > title			
year	title	nominaciones	productora
1939	The Wizard of Oz	13	Warner Bros. Pi...
1961	The Guns of N...	11	Columbia Pict...
1962	Lawrence of Ar...	14	Columbia Pict...
1965	The Great Race	12	Warner Home ...
1966	The Sand Pebb...	10	Twentieth Cent...
1967	Doctor Dolittle	14	20th Century F...
1972	Deliverance	10	Warner Bros.
1972	The Poseidon ...	13	20th Century F...
1972	Travels with M...	11	
1975	Barry Lyndon	11	Warner Bros.
1975	Jaws	14	Universal Pictu...

12)

Base de datos: sample\_mflix

Colección: movies

Obtener el número total de horas de Thriller.

```
db.movies.aggregate([
  {$match: {"genres": "Thriller"}}, // Filtra con esta condición
  {$group: { // Agrupa y calcula la suma de los valores del campo runtime para obtener el número total de horas de las películas de Thriller
    _id: null,
    HorasTotales: {$sum: "$runtime"}},
  {$project: {_id: 0, HorasTotales: 1}}
])
```

movies	
HorasTotales	
253309	

13)

Base de datos: sample\_mflix

Colección: movies y comments

Obtener un listado de las 20 películas con mayor número de comentarios que hayan sido publicadas en el año 2015.

```

db.movies.aggregate([
  {$match: {year: 2015}},
  {$sort: {num_mflix_comments: -1}},
  {$limit: 20},
  {$project: {_id: 0, title: 1, year: 1, num_mflix_comments: 1}}
])

```

movies > title		
num_mflix_comme...	title	year
3	45 Years	2015
3	Tired Moonlight	2015
3	3 1/2 Minutes, ...	2015
3	The Shaman	2015
3	The Saboteurs	2015
2	Polycarp	2015
2	This Isn't Funny	2015
2	The Adderall Di...	2015
2	Example	2015

14)

Base de datos: sample\_training

Colección: grades

Obtener la nota media por clase del estudiante cuyo id es 2378.

use sample\_training

```

db.grades.aggregate([
  {$unwind: "$scores"}, // Deconstruye el array de scores
  {$match: {student_id: 2378}}, // Se queda con todas las notas del alumno
  // {$match: {$and: [{student_id: 2378}, {"scores.type": "exam"}]}}, // Si consideramos solo la media de los de tipo examen
  {$group: {_id: "$class_id", media: {$avg: "$scores.score"}}}
])

```

grades > media	
_id	media
339.0	41.2635468290...
22.0	41.4511560870...
25.0	46.2488928341...
204.0	46.0767135334...
459.0	41.0900543661...
232.0	77.4993634210...
241.0	51.5881169469...
401.0	58.4705463744...
111.0	64.2425879379...
307.0	49.6485707932...

15)

Base de datos: sample\_training

Colección: grades

Encontrar el estudiante con mejor nota media en la clase con id 122.

```
db.grades.aggregate([
  {$unwind: "$scores"}, // Deconstruye el array de scores
  {$match: {class_id: 122}}, // Selecciona la clase 122
  {$group: {_id: "$student_id", media: {$avg: "$scores.score"}}},
  {$sort: {media: -1}}, // Ordena descendientemente para quedarse con el primero
  {$limit: 1}
])
```

grades > media	
_id	media
123 8493.0	123 94.6718908756...

16)

Base de datos: sample\_training

Colección: grades

Obtener un listado de notas medias por tipo de puntuación, para la clase con id 122.

```
db.grades.aggregate([
  {$unwind: "$scores"}, // Deconstruye el array de scores
  {$match: {class_id: 122}}, // Filtra (clase con class_id= 122)
  {$group: { // Agrupa por el campo scores.type, y calcula la nota media por cada tipo de puntuación
    _id: "$scores.type",
    media: {$avg: "$scores.score"}}}
])
```

grades > media	
_id	media
exam	123 48.4131344908...
homework	123 51.0373244247...
quiz	123 50.7919484843...