Introducción

Aggregations permite realizar queries anidadas que incluyan operaciones, es decir, procesan registros de datos de una o varias colecciones y devuelven resultados calculados.

Las operaciones de agregación permiten realizar varias operaciones que se irán ejecutando una detrás de otra siguiendo el orden especificado.

Sintaxis
db.collection.aggregate (canalización, opciones)

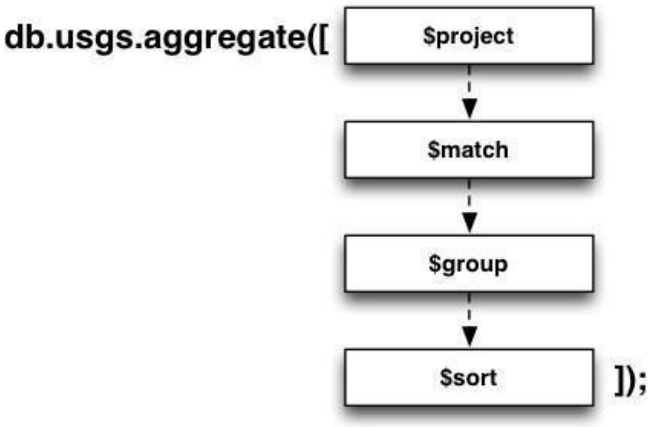
Parámetros

Parámetro	Detalles
tubería	matriz (una secuencia de operaciones o etapas de agregación de datos)
opciones	documento (opcional, disponible solo si la canalización está presente como una matriz)

MongoDB proporciona tres formas de realizar la agregación: el canal o túnel de agregación, la función de reducción de mapas (no muy eficiente) y los métodos de agregación de propósito único.
Manual de Mongo https://docs.mongodb.com/manual/aggregation/

Estructura

Nosotros nos vamos a centrar en la manera más óptima de hacer una agregación de tipo "pipeline": diferentes etapas, donde cada una toma la salida de la anterior.



Las fases se pueden repetir:

Operado	Descripción	Cardinalida
$project	Proyección de campos, es decir, propiedades en las que estamos interesados. También nos permite modificar un documento, o crear un subdocumento (*reshape*)	1:1
$match	Filtrado de campos, similar a *where*	N:1
$group	Para agrupar los datos, similar a *group by*	N:1
$sort	Ordenar	1:1
$skip	Saltar	N:1
$limit	Limitar los resultados	N:1
$unwind	Separa los datos que hay dentro de un array	1:N

Hay algunas operaciones como $project, $match, $limit, $skip que tienen el mismo funcionamiento que hemos visto en una query find().

$project
Cambia la forma de cada documento en la secuencia, por ejemplo, agregando nuevos campos o eliminando campos existentes. Para cada documento de entrada, genera un documento.

$match
Filtra el flujo de documentos para permitir que solo los documentos coincidentes pasen sin modificar a la siguiente etapa de canalización. $match utiliza consultas estándar de MongoDB. Para cada documento de entrada, genera un documento (una coincidencia) o cero documentos (ninguna coincidencia).

$redact
Cambia la forma de cada documento en la secuencia restringiendo el contenido de cada documento en función de la información almacenada en los propios documentos. Incorpora la funcionalidad de $project y $match. Se puede utilizar para implementar la redacción a nivel de campo. Para cada documento de entrada, genera uno o cero documentos.

$limit
Pasa los primeros n documentos sin modificar a la canalización donde n es el límite especificado. Para cada documento de entrada, genera un documento (para los primeros n documentos) o cero documentos (después de los primeros n documentos).

$skip
Omite los primeros n documentos donde n es el número de omisión especificado y pasa los documentos restantes sin modificar a la canalización. Para cada documento de entrada, genera cero documentos (para los primeros n documentos) o un documento (si es después de los primeros n documentos).

$unwind
Deconstruye un campo de matriz a partir de los documentos de entrada para generar un documento para cada elemento. Cada documento de salida reemplaza la matriz con un valor de elemento. Para cada documento de entrada, genera n documentos donde n es el número de elementos de la matriz y puede ser cero para una matriz vacía.

$group
Agrupa los documentos de entrada por una expresión de identificador especificada y aplica la (s) expresión (es) del acumulador, si se especifica, a cada grupo. Consume todos los documentos de entrada y genera un documento por cada grupo distinto. Los documentos de salida solo contienen el campo identificador y, si se especifica, campos acumulados.

$sample
Selecciona aleatoriamente el número especificado de documentos de su entrada.

$sort
Reordena el flujo de documentos mediante una clave de clasificación especificada. Solo cambia el orden; los documentos permanecen sin modificar. Para cada documento de entrada, genera un documento.

$geoNear
Devuelve un flujo ordenado de documentos según la proximidad a un punto geoespacial. Incorpora la funcionalidad de $match, $sort y $limit para los datos geoespaciales. Los documentos de salida incluyen un campo de distancia adicional y pueden incluir un campo de identificación de ubicación.

$lookup
Realiza una unión externa a otra colección en la misma base de datos para filtrar los documentos de la colección "referenciada" para su procesamiento.
Hay cuatro campos obligatorios:

- from: La colección que se usará para la búsqueda en la misma base de datos
- localField: el campo de la colección principal que se puede utilizar como identificador único en la fromcolección.
- foreignField: el campo de la fromcolección que se puede utilizar como identificador único en la colección principal.
- as: El nombre del nuevo campo que contendrá los documentos coincidentes de la fromcolección.

$out
Escribe los documentos resultantes de la canalización de agregación en una colección. Para utilizar la etapa $out, debe ser la última etapa del proceso.

$indexStats
Devuelve estadísticas sobre el uso de cada índice para la colección.

Optimización

Mongo utiliza los índices en las operaciones de agregación solo para los operadores $match y $sort.

Mongo **optimiza automáticamente** las operaciones de agregación para que sean más eficientes siguiendo unas pautas:

- Ejecuta siempre antes un $match que un $sort, lo adelanta para reducir el número de registros con el que seguir operando. Y siempre usando los índices.
- Si se encuentra un $skip antes que un $limit, Mongo los reordena al revés cambiando $limit por $limit +$skip
- Cuando Mongo detecta que dos comandos pueden fusionarse lo hace internamente (ej $limit seguidos : $limit(5) y $limit(2) hace directamente $limit(2))
  (ej $skip seguidos: $skip(5) y $skip(2) hace directamente $skip(7)).
- Cuando tenemos $project seguido de $skip o $limit, Mongo adelanta a los segundos para que la proyección sea sobre menos resultados o para que si hay un $sort delante pueda fusionarse.

Como programadores también podemos seguir una serie de pautas para optimizar nuestras agregaciones:

- Cuando tenemos dos $match seguidos, es má eficaz realizar la operación en uno sólo y utilizar el operador lógico $and.
- Cuando ejecutemos un $lookup seguido de un $unwind podemos optimizarlo poniendo el $unwind dentro del propio $lookup.

EJEMPLO

- Crear algunos datos ficticios:
  db.employees.insertOne({"name":"Adma","dept":"Admin","languages":["german","french","english","hindi"],"age":30,"totalExp":10});
  db.employees.insertOne({"name":"Anna","dept":"Admin","languages":["english","hindi"],"age":35,"totalExp":11});
  db.employees.insertOne({"name":"Bob","dept":"Facilities","languages":["english","hindi"],"age":36, "totalExp":14});
  db.employees.insertOne({"name":"Cathy","dept":"Facilities","languages":["hindi"],"age":31, "totalExp":4});
  db.employees.insertOne({"name":"Mike","dept":"HR","languages":["english", "hindi", "spanish"],"age":26, "totalExp":3});
  db.employees.insertOne({"name":"Jenny","dept":"HR","languages":["english", "hindi", "spanish"],"age":25, "totalExp":3});

- Obtener los datos de los empleados del departamento Admin.
  db.employees.aggregate([{$match:{dept:"Admin"}}]);
  Output:
  { "_id" : ObjectId("54982fac2e9b4b54ec384a0d"), "name" : "Adma", "dept" : "Admin", "languages": ["german", "french", "english", "hindi"], "age" : 30, "totalExp" : 10 }

{ "_id" : ObjectId("54982fc92e9b4b54ec384a0e"), "name" : "Anna", "dept" : "Admin", "languages": ["english", "hindi"], "age" : 35, "totalExp" : 11 }

- Mostrar sólo los datos de id, nombre y departamento de los empleados del departamento Admin.
  db.employees.aggregate([{$match:{dept:"Admin"}}, {$project:{"name":1, "dept":1}}]);
  Output:
  { "_id" : ObjectId("54982fac2e9b4b54ec384a0d"), "name" : "Adma", "dept" : "Admin" }
  { "_id" : ObjectId("54982fc92e9b4b54ec384a0e"), "name" : "Anna", "dept" : "Admin" }

- Mostrar sólo el nombre de los empleados.
  db.employees.aggregate({$project: {'_id':0, 'name': 1}})
  Output:
  { "name" : "Adma" }
  { "name" : "Anna" }
  { "name" : "Bob" }
  { "name" : "Cathy" }
  { "name" : "Mike" }
  { "name" : "Jenny" }

- Mostrar los departamentos existentes.
  Aquí los documentos se agrupan por el valor del campo "dept".
  db.employees.aggregate([{$group:{"_id":"$dept"}}]);
  { "_id" : "HR" }
  { "_id" : "Facilities" }
  { "_id" : "Admin" }

- Listado de id, nombre y departamento de los empleados del departamento de Admin ordenados alfabéticamente por nombre.
  db.employees.aggregate([{$match:{dept:"Admin"}}, {$project:{"name":1, "dept":1}}, {$sort: {name: 1}}]);
  Output:
  { "_id" : ObjectId("57ff3e553dedf0228d4862ac"), "name" : "Adma", "dept" : "Admin" }
  { "_id" : ObjectId("57ff3e5e3dedf0228d4862ad"), "name" : "Anna", "dept" : "Admin" }

  db.employees.aggregate([{$match:{dept:"Admin"}}, {$project:{"name":1, "dept":1}}, {$sort: {name: -1}}]);
  Output:
  { "_id" : ObjectId("57ff3e5e3dedf0228d4862ad"), "name" : "Anna", "dept" : "Admin" }
  { "_id" : ObjectId("57ff3e553dedf0228d4862ac"), "name" : "Adma", "dept" : "Admin" }

- Listado de id, nombre y departamento de los empleados del departamento de Admin, a partir del segundo, ordenados descendentemente por nombre.
  db.employees.aggregate([{$match:{dept:"Admin"}}, {$project:{"name":1, "dept":1}}, {$sort: {name: -1}}, {$skip:1}]);
  Output:
  { "_id" : ObjectId("57ff3e553dedf0228d4862ac"), "name" : "Adma", "dept" : "Admin" }

- Visualiza el id, nombre y departamento del primer empleado del departamento de Admin ordenados descendentemente por nombre.
  db.employees.aggregate([{$match:{dept:"Admin"}}, {$project:{"name":1, "dept":1}}, {$sort:{name: -1}}, {$limit:1}]);
  Output:
  { "_id" : ObjectId("57ff3e5e3dedf0228d4862ad"), "name" : "Anna", "dept" : "Admin" }

  Obtener datos de muestra aleatoria de 2 elementos
  db.emplyees.aggregate({ $sample: { size:2 } })

  NOTA: Hasta ahora hemos hecho con Agregaciones lo mismo que hacíamos antes con find, distinct, sort, skip y limit.

Acumuladores

Los acumuladores están disponibles en los escenarios $group y $project.
Los acumuladores, cuando se utilizan en la etapa $group, mantienen su estado (por ejemplo, totales, máximos, mínimos y datos relacionados) a medida que los documentos avanzan en el proceso.Los acumuladores toman como entrada una sola expresión, evaluando la expresión una vez para cada documento de entrada y mantienen su etapa para el grupo de documentos que comparten la misma clave de grupo.

Cuando se usa en el escenario $project, los acumuladores no mantienen su estado. Los acumuladores toman como entrada un solo argumento o múltiples argumentos.

$sum
Devuelve una suma de valores numéricos. Ignora los valores no numéricos.
Disponible en ambas etapas $group y $project.

$avg
Devuelve un promedio de valores numéricos. Ignora los valores no numéricos.
Disponible en ambas etapas $group y $project.

$first
Devuelve un valor del primer documento para cada grupo. El orden solo se define si los documentos están en un orden definido.
Disponible sólo con $group.

$last
Devuelve un valor del último documento para cada grupo. El orden solo se define si los documentos están en un orden definido.
Disponible sólo con $group.

$max
Devuelve el valor de expresión más alto para cada grupo.
Disponible en ambas etapas $group y $project.

$min
Devuelve el valor de expresión más bajo para cada grupo.
Disponible en ambas etapas $group y $project.

$push
Devuelve una matriz de valores de expresión para cada grupo.
Disponible sólo con $group.

$addToSet
Devuelve una matriz de valores de expresión únicos para cada grupo. El orden de los elementos de la matriz no está definido.
Disponible sólo con $group.

$stdDevPop
Devuelve la desviación estándar de la población de los valores de entrada.
Disponible en ambas etapas $group y $project.

$stdDevSamp
Devuelve la desviación estándar de la muestra de los valores de entrada.

EJEMPLO: Practica con algunos de los acumuladores

- Calcula la suma de las edades de todos los empleados.
  db.employees.aggregate([{$group:{"_id":null, "totalAge":{$sum:"$age"}}}]);
  Output:
  { "_id" : null, "totalAge" : 183 }

- Cuenta el número de empleados de cada departamento.
  db.employees.aggregate([{$group:{"_id":"$dept", "noOfDept":{$sum:1}}}]);
  Output:
  { "_id" : "HR", "noOfDept" : 2 }
  { "_id" : "Facilities", "noOfDept" : 2 }
  { "_id" : "Admin", "noOfDept" : 2 }

- Calcula el promedio del total de experiencia por departamento.
  db.employees.aggregate([{$group:{"_id":"$dept", "avgExp":{$avg:"$totalExp"}}}]);
  Output:
  { "_id" : "HR", "avgExp" : 3 }
  { "_id" : "Facilities", "avgExp" : 9 }
  { "_id" : "Admin", "avgExp" : 10.5 }

- Encuentra la experiencia mínima en cada departamento.
  db.employees.aggregate([{$group:{"_id":"$dept", "minExp":{$min:"$totalExp"}}}]);
  Output:
  { "_id" : "HR", "minExp" : 3 }
  { "_id" : "Facilities", "minExp" : 4 }
  { "_id" : "Admin", "minExp" : 10 }

- Encuentra la experiencia máxima en cada departamento.
  db.employees.aggregate([{$group:{"_id":"$dept", "maxExp":{$max:"$totalExp"}}}]);
  Output:
  { "_id" : "HR", "maxExp" : 3 }
  { "_id" : "Facilities", "maxExp" : 14 }
  { "_id" : "Admin", "maxExp" : 11 }

- Obtención del valor del campo nombre del primer y último empleado de cada grupo según edad.
  db.employees.aggregate([{$group:{"_id":"$age", "lasts":{$last:"$name"},
  "firsts":{$first:"$name"}}}]);
  Output:
  { "_id" : 25, "lasts" : "Jenny", "firsts" : "Jenny" }
  { "_id" : 26, "lasts" : "Mike", "firsts" : "Mike" }
  { "_id" : 35, "lasts" : "Cathy", "firsts" : "Anna" }
  { "_id" : 30, "lasts" : "Adma", "firsts" : "Adma" }

- Obtener la experiencia máxima y mínima por departamento.
  db.employees.aggregate([{$group:{"_id":"$dept", "maxExp":{$max:"$totalExp"},
  "minExp":{$min: "$totalExp"}}}]);
  Output:
  { "_id" : "HR", "maxExp" : 3, "minExp" : 3 }
  { "_id" : "Facilities", "maxExp" : 14, "minExp" : 4 }
  { "_id" : "Admin", "maxExp" : 11, "minExp" : 10 }

- Mostrar las distintas edades de los empleados (con y sin repeticiones).
  db.employees.aggregate([{$group:{"_id":"dept", "arrPush":{$push:"$age"}, "arrSet":
  {$addToSet:"$age"}}}]);
  Output:
  { "_id" : "dept", "arrPush" : [30, 35, 36, 35, 26, 25], "arrSet" : [25, 26, 35, 30] }

- Mostrar todos los datos de Adma por cada lenguaje, luego podríamos hacer una agregación adicional basada en esos valores.
  db.employees.aggregate([{$match:{"name":"Adma"}}, {$unwind:"$languages"}]);
  Output:
  { "_id" : ObjectId("54982fac2e9b4b54ec384a0d"), "name" : "Adma", "dept" : "HR",
  "languages" : "german", "age" : 30, "totalExp" : 10 }
  { "_id" : ObjectId("54982fac2e9b4b54ec384a0d"), "name" : "Adma", "dept" : "HR",
  "languages" : "french", "age" : 30, "totalExp" : 10 }
  { "_id" : ObjectId("54982fac2e9b4b54ec384a0d"), "name" : "Adma", "dept" : "HR",
  "languages" : "english", "age" : 30, "totalExp" : 10 }
  { "_id" : ObjectId("54982fac2e9b4b54ec384a0d"), "name" : "Adma", "dept" : "HR",
  "languages" : "hindi", "age" : 30, "totalExp" : 10 }

Operadores de comparación

Las expresiones de comparación devuelven un valor booleano, excepto $cmp que devuelve un número.

$cmp
Devuelve: 0 si los dos valores son equivalentes, 1 si el primer valor es mayor que el segundo y -1 si el primer valor es menor que el segundo.

$eq
Devuelve true si los valores son equivalentes.

$gt
Devuelve true si el primer valor es mayor que el segundo.

$gte
Devuelve true si el primer valor es mayor o igual que el segundo.

$lt
Devuelve true si el primer valor es menor que el segundo.

$lte
Devuelve true si el primer valor es menor o igual que el segundo.

$ne
Devuelve true si los valores no son equivalentes.

$in
Devuelve true si un valor está contenido en un array.

$nin
Devuelve true si un valor no está contenido en un array.

Operador condicional

$cond
Realiza una cosa u otra dependiendo de si se cumple o no la condición especificada
Formato: $cond ((condición), resultado si true, resultado si false)
si no queremos que haga nada en alguno de los resultados usamos “$noval”.

Operadores booleanos

Las expresiones booleanas evalúan sus expresiones de argumentos como booleanos y devuelven un booleano como resultado.

Además del valor booleano false, expresión se evalúa booleanos como false los siguientes: null, 0 y undefined valores. La expresión booleana evalúa todos los demás valores como true, incluidos los valores numéricos y matrices distintos de cero.

$and
Devuelve true solo cuando todas sus expresiones se evalúan como true. Acepta cualquier número de expresiones de argumento.

$or
Devuelve true cuando cualquiera de sus expresiones se evalúa como true. Acepta cualquier número de expresiones de argumento.

$not
Devuelve el valor booleano opuesto a la expresión de su argumento. Acepta una expresión de un solo argumento.

EJEMPLO: Practica con algunos de los operadores

- Listado de los empleados del departamento Admin, indicando id, nombre, departamento y si es mayor o no de 30.
  db.employees.aggregate([{$match:{dept:"Admin"}}, {$project:{"name":1, "dept":1, age: {$gt: ["$age", 30]}}}]);
  Output:
  { "_id" : ObjectId("57ff3e553dedf0228d4862ac"), "name" : "Adma", "dept" : "Admin", "age" : false }
  { "_id" : ObjectId("57ff3e5e3dedf0228d4862ad"), "name" : "Anna", "dept" : "Admin", "age" : true }

- Listado de los campos id, nombre, departamento de los empleados del departamento Admin que sean mayores de 30.
  db.employees.aggregate([{$match:{dept:"Admin", age: {$gt:30}}}, {$project:{"name":1, "dept":1}}]);
  Output:
  { "_id" : ObjectId("57ff3e5e3dedf0228d4862ad"), "name" : "Anna", "dept" : "Admin" }

- Listado de los empleados del departamento Admin, indicando id, nombre, departamento y si su edad está comprendida entre los 31 y 35.
  db.employees.aggregate([{$match:{dept:"Admin"}}, {$project:{"name":1, "dept":1, age: { $and: [{ $gt: ["$age", 30] }, { $lt: ["$age", 36] }] }}}]);
  Output:
  { "_id" : ObjectId("57ff3e553dedf0228d4862ac"), "name" : "Adma", "dept" : "Admin", "age" :

false }
{ "_id" : ObjectId("57ff3e5e3dedf0228d4862ad"), "name" : "Anna", "dept" : "Admin", "age" :
true }

- Listado de los campos id, nombre, departamento de los empleados del departamento Admin cuya edad esté comprendida entre los 31 y 35.
db.employees.aggregate([{$match:{dept:"Admin", $and: [{age: { $gt: 30 }}, {age: {$lt: 36 }}] }}, {$project:{"name":1, "dept":1, age: { $and: [{ $gt: ["$age", 30] }, { $lt: ["$age", 36] }] }}}]);
Output:
{ "_id" : ObjectId("57ff3e5e3dedf0228d4862ad"), "name" : "Anna", "dept" : "Admin", "age" :
true }

Operadores aritméticos

Los operadores aritméticos, los utilizamos en otros pipelines como $project.

Los operadores que podemos utilizar son:

$add
Realiza la suma de un array de números.

$divide
Divide dos números.

$mod
A partir de dos números calcula el resto producido al dividir el primero entre el segundo.

$multiply
Multiplica dos números.

$substract
A partir de dos números realiza la resta.

Operadores de cadenas

$concat
Concatena cualquier número de cadenas.

$substr
Devuelve una subcadena de una cadena, comenzando en una posición de índice especificada hasta una longitud especificada. Acepta tres expresiones como argumentos: el

primer argumento debe resolverse en una cadena y el segundo y tercer argumentos deben resolverse en números enteros.

$toLower
Convierte una cadena a minúsculas. Acepta una expresión de un solo argumento.

$toUpper
Convierte una cadena en mayúsculas. Acepta una expresión de un solo argumento.

$strcasecmp
Realiza una comparación de cadenas que no distingue entre mayúsculas y minúsculas y devuelve: 0 si dos cadenas son equivalentes, 1 si la primera cadena es mayor que la segunda y -1 si la primera cadena es menor que la segunda.

Operadores de array

$arrayElemAt
Devuelve el elemento en el índice de matriz especificado.

$concatArrays
Concatena matrices para devolver la matriz concatenada.

$filter
Selecciona un subconjunto de la matriz para devolver una matriz con solo los elementos que coinciden con la condición del filtro.

$isArray
Determina si el operando es una matriz. Devuelve un booleano.

$size
Devuelve el número de elementos de la matriz. Acepta una sola expresión como argumento.

$slice
Devuelve un subconjunto de una matriz.

Operadores de fecha

$dayOfYear
Devuelve el día del año para una fecha como un número entre 1 y 366 (año bisiesto).

$dayOfMonth
Devuelve el día del mes para una fecha como un número entre 1 y 31.

$dayOfWeek
Devuelve el día de la semana para una fecha como un número entre 1 (domingo) y 7 (sábado).

$year
Devuelve el año de una fecha como un número (por ejemplo, 2014).

$month
Devuelve el mes de una fecha como un número entre 1 (enero) y 12 (diciembre).

$week
Devuelve el número de semana de una fecha como un número entre 0 (la semana parcial que precede al primer domingo del año) y 53 (año bisiesto).

$hour
Devuelve la hora de una fecha como un número entre 0 y 23.

$minute
Devuelve el minuto de una fecha como un número entre 0 y 59.

$second
Devuelve los segundos de una fecha como un número entre 0 y 60 (segundos intercalares).

$millisecond
Devuelve los milisegundos de una fecha como un número entre 0 y 999.

$dateToString
Devuelve la fecha como una cadena formateada.