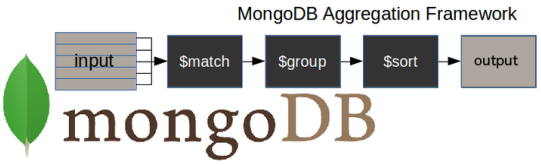
Proceso de Agregación en MongoDB:

La agregación en MongoDB es el proceso de transformar y combinar datos de diferentes documentos en colecciones en uno o más resultados. Se utiliza la agregación para realizar operaciones de transformación de datos similares a la función GROUP BY en SQL.



Sintaxis de Pipeline:

La agregación en MongoDB se realiza utilizando un conjunto de etapas que se ejecutan secuencialmente, llamado pipeline. Cada etapa en el pipeline realiza una operación específica en los documentos de entrada y pasa los resultados a la siguiente etapa.

```
db.collection.aggregate([
  { <etapa1> },
  { <etapa2> },
  { <etapa3> },  ...])
```

Etapas de Agregación Comunes:

$match: Filtra los documentos para procesar solo los que cumplan con ciertos criterios.

$project: Cambia la estructura de un documento y puede incluir o excluir campos.

$group: Agrupa documentos por un campo específico y realiza operaciones de agregación en cada grupo.

$addFields: Agrega nuevos campos a los documentos existentes.

$lookup: Realiza una operación de unión (join) entre documentos de dos colecciones.

$sort: Ordena los documentos en función de un campo específico.

$limit: Limita el número de documentos que pasan a la siguiente etapa.

$unwind: Descompone un campo de matriz en varios documentos, uno por cada valor de matriz.

$out: Escribe los resultados de la agregación en una nueva colección.

Operadores de Agregación de Group:

$sum: Suma los valores especificados.

$avg: Calcula el promedio de los valores especificados.

$min: Encuentra el valor mínimo de los valores especificados.

$max: Encuentra el valor máximo de los valores especificados.

$push: Agrega valores a un array.

$addToSet: Agrega valores únicos a un array.

$first: Devuelve el primer valor de un grupo.

$last: Devuelve el último valor de un grupo.

$count: Cuenta el número de documentos en un grupo.

Ejemplo de Pipeline de agregación:

```
db.collection.aggregate([
   { $match: { field1: { $gte: 100 } } },
   { $project: { _id: 0, field1: 1, field2: 1 } },
   { $group: { _id: "$field1", total: { $sum: "$field2" } } },
   { $sort: { total: -1 } },
   { $limit: 10 },
   { $out: "nueva_coleccion" }
])
```

$match: Filtra documentos con "field1" mayor o igual a 100.

$project: Proyecta solo los campos "field1" y "field2" sin el "_id".

$group: Agrupa documentos por "field1" y calcula la suma de "field2" para cada grupo.

$sort: Ordena los documentos por el campo "total" en orden descendente.

$limit: Limita los resultados a 10 documentos.

$out: Escribe los resultados de la agregación en una nueva colección llamada "nueva_coleccion".

Ejemplo 1: Usando $match y $group

Supongamos que tenemos una colección de ventas con documentos que tienen el siguiente formato:

{ "_id": 1, "producto": "A", "cantidad": 10, "precio_unitario": 5 },

{ "_id": 2, "producto": "B", "cantidad": 20, "precio_unitario": 10 },

{ "_id": 3, "producto": "A", "cantidad": 15, "precio_unitario": 6 },

{ "_id": 4, "producto": "C", "cantidad": 5, "precio_unitario": 8 }

Queremos encontrar el total de ventas para cada producto. Podemos usar la agregación:

```
db.ventas.aggregate([
  { $group: { _id: "$producto", total_ventas: { $sum: { $multiply: ["$cantidad", "$precio_unitario"] } } } }
])
```

- Usamos $group para agrupar los documentos por el campo "producto".
- Usamos $sum para calcular el total de ventas para cada grupo, que es el resultado de multiplicar la cantidad por el precio unitario para cada documento.

Ejemplo 2: usando $match, $group

Supongamos que sobre la colección ventas queremos calcular el total de ventas, la cantidad total de productos vendidos y el promedio de precio unitario por producto para el mes de enero de 2024.

```
db.ventas.aggregate([
  { $match: { fecha: { $gte: ISODate("2024-01-01"), $lt: ISODate("2024-02-01") } } },  // Filtrar 01/24
  { $group: { _id: "$producto",
        total_ventas: { $sum: { $multiply: ["$cantidad", "$precio_unitario"] } },  // Total por producto
        total_cantidad: { $sum: "$cantidad" },  // Total de productos vendidos por producto
        promedio_precio_unitario: { $avg: "$precio_unitario" }  // Promedio de precio por producto
      } }])
```

Guía útil: Juan Roy Couto. Soporte documental

- Usamos $match para filtrar los documentos por la fecha de enero de 2024.
- Luego, usamos $group para agrupar los documentos por el campo "producto" y realizar operaciones de agregación en cada grupo:
- $sum para calcular el total de ventas multiplicando la cantidad por el precio unitario.
- $sum para calcular la cantidad total de productos vendidos.
- $avg para calcular el promedio de precio unitario.

Ejemplo 3: Usando $lookup

Supongamos que tenemos dos colecciones, una para clientes y otra para pedidos. Queremos obtener los pedidos junto con la información del cliente al que pertenecen:

```
db.pedidos.aggregate([
  { $lookup: { from: "clientes", localField: "cliente_id", foreignField: "_id", as: "cliente_info" } }
])
```

- Usamos $lookup para realizar una operación de unión entre la colección de pedidos y la colección de clientes.
- from especifica la colección con la que queremos unirnos.
- localField especifica el campo en la colección de pedidos que se utilizará para la unión.
- foreignField especifica el campo en la colección de clientes que se utilizará para la unión.