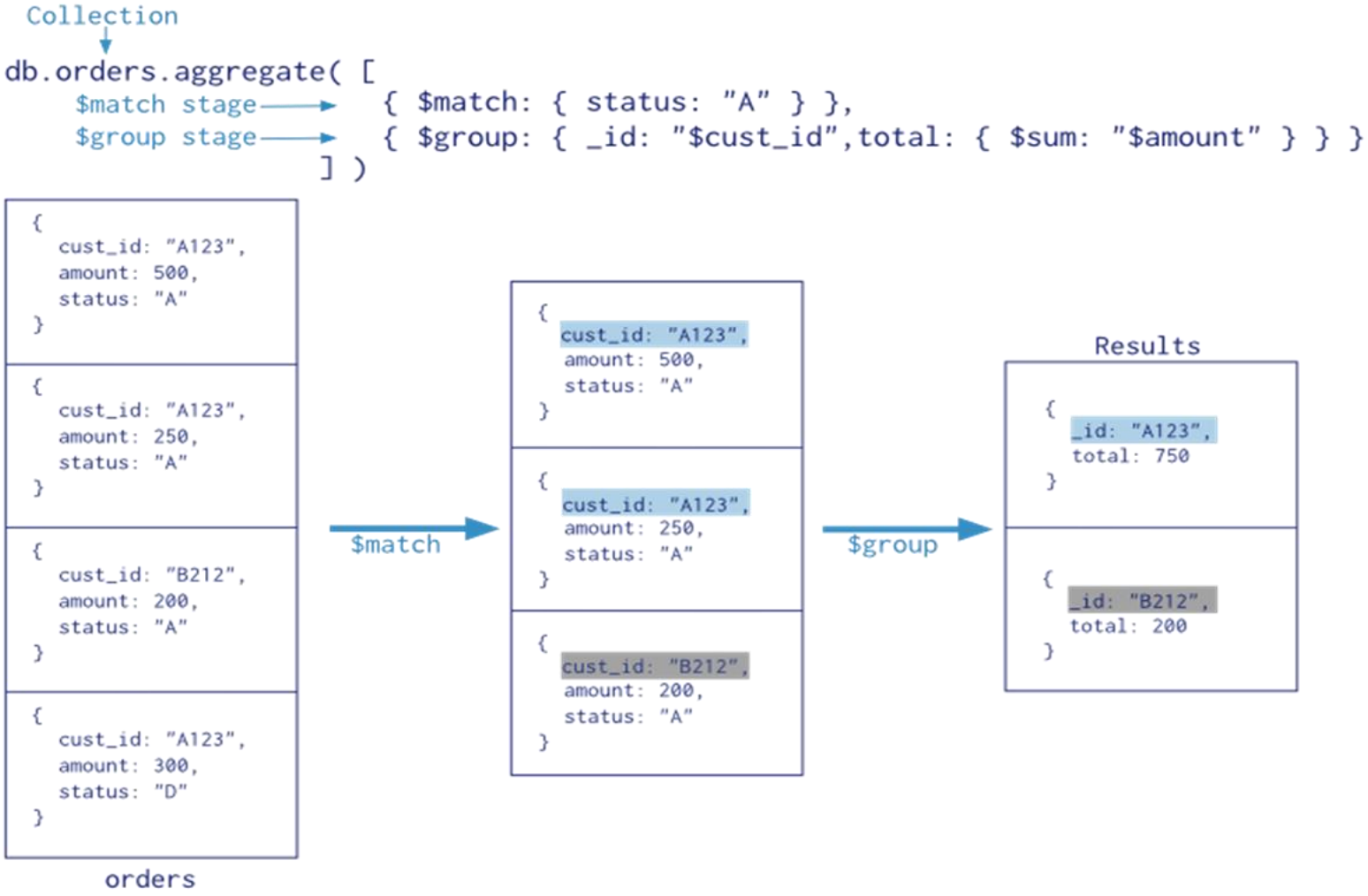
NoSQL

Carrera Programador full-stack

Introduccion

MongoDB - Aggregation Pipeline

- Los docs entran a un multi-stage pipeline que genera documentos con los agregados



MongoDB - Aggregation Pipeline

El **Aggregation Pipeline** es una forma poderosa en MongoDB de transformar, filtrar, agrupar y procesar datos dentro de una colección.
Consiste en **etapas encadenadas**, como si fueran pasos en una fábrica, cada una operando sobre los resultados de la anterior.
Se usa con:

```
db.ventas.aggregate([  etapas  ])
```

MongoDB - Aggregation Pipeline

Etapa	Función principal
$match	Filtra documentos (como un WHERE en SQL)
$project	Muestra/oculta campos, o crea nuevos
$set / $addFields	Agrega campos calculados o nuevos
$group	Agrupa documentos por algún campo, permitiendo sumas, promedios, etc.
$sort	Ordena los documentos según uno o varios campos

Ejemplos - Aggregation Pipeline

Generamos la colección “ventas” para practicar, con los siguientes campos:

Producto, categoría, precio, cantidad, vendedor, fecha

VAMOS A VER CON EJEMPLOS!

Ejemplos - Aggregation Pipeline

$match funciona como filtro, podemos comparar con WHERE de SQL:

- Filtramos todas las ventas que sean de la categoría Tecnologia:

```
db.ventas.aggregate([
  {
    $match: { categoria: "Tecnología" }
  }
])
```

$project muestra/oculta o crea nuevos campos:

- Mostrar solo producto, cantidad y precio (sin _id):

```
db.ventas.aggregate([
  {
    $project: {
      _id: 0,
      producto: 1,
      cantidad: 1,
      precio: 1
    }
  }
])
```

Ejemplos - Aggregation Pipeline

$set agrega campos modificados o nuevos

- Creamos un campo que calcule el precio total (precio * cantidad)

```
db.ventas.aggregate([
  {
    $set: {
      total: { $multiply: ["$precio", "$cantidad"] }
    }
  }
])
```

$group agrupa por campos:

- Monto total de ventas en una categoría:

```
db.ventas.aggregate([
  {
    $group: {
      _id: "$categoria",
      totalVentacategorias: { $sum: { $multiply: ["$precio", "$cantidad"] } }
    }
  }
])
```

Ejemplos - Aggregation Pipeline

$sort ordenar en orden ascendente o descendente

- Ordenamos los productos en orden descente en cantidad de ventas:

```
db.ventas.aggregate([
  {
    $sort: { cantidad: -1 }
  }
])
```

Tambien podemos combinarlos. Por ejemplo: Mostrar solo ventas que son mayores a $500. Vamos a usar, $set y $match

```
db.ventas.aggregate([
  {
    $set: {
      total: { $multiply: ["$precio", "$cantidad"] }
    }
  },
  {
    $match: {
      total: { $gt: 500 }
    }
  }
])
```

Practica - Aggregation Pipeline

•Listá todas las ventas correspondientes a la categoría "Hogar".
•Mostrá únicamente el nombre del producto y su precio.
•Ordená los productos por precio de forma descendente.
•Agregá un campo nuevo llamado total que represente el total de la venta (precio por cantidad).
•Mostrá únicamente las ventas cuyo total sea mayor a $1000.
•Listá el producto, categoría y un nuevo campo llamado iva que represente el 21% del total de la venta.
•Mostrá todas las ventas realizadas por el vendedor "Juan", ordenadas por cantidad vendida.
•Calculá el total de ventas por cada categoría.
•Mostrá la cantidad total de productos vendidos por cada vendedor.
•Listá las categorías que hayan generado más de $2000 en ventas.
•Mostrá la venta con mayor cantidad de productos vendidos.
•Generá un informe que incluya el producto, la categoría, el total vendido, el valor del IVA (21%) y el precio final (total más IVA).
•Agrupá las ventas por día y mostrales la cantidad de ventas realizadas en cada fecha.
•Calculá el ticket promedio (monto total dividido la cantidad de ventas) para cada vendedor.