# TRYBE
# Modulo III – Back-end

## Bloco 25 – MongoDB - Aggregation Framework

### 1) Aggregation Framework - Parte 1

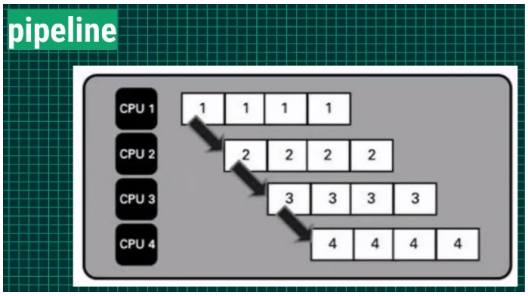
O que é :
Recurso nativo do MongoDB escrito em **C++**.

O que fazem Afs:
Processam dados vindo de diferentes documents ou collections e retornam resultado calculado.

Como :
MongoDB não permite que operações de agrupamento sejam feitas com método find().
MongoDB fornece três caminhos para executar operações de agregação: **aggregation pipeline** (o que veremos aqui)**, map-reduce function e single purpose aggregation methods**.

#### Aggregation Pipeline

**Pipeline ou funil que vai de estágio em estágio até apresentar resultado agregado.**




```
db.orders.aggregate([
  { $match: { status: "A" } },  // first stage
  { $group: { _id: "$cust_id", total: { $sum:
"$amount" } } } // second stage
]);
```


Nestes **estágios ocorrem transformações de documentos, via uso de filtros, agrupamento, ordenação,** operadores**, índices**, etc.

#### $match

**Filtra** documentos do mesmo jeito que filtros no find. Primeira etapa para melhorar performance.

```
db.collection.aggregate(
  [{ $match : { campo : "bla" } }]
);
```

## $limit

**Limita** os resultados ou seja o número de documentos que será passado para o próximo estágio.
*db.collection.aggregate([{ $limit : 5 }]);*

## $lookup

**Junta (join) dados** de uma ou mais collections.
**Resultado: um elemento do tipo array** é adicionado a cada documento da coleção de entrada, contendo os documentos que deram "match" na coleção com a qual se faz o "join".

4 parâmetros: **from, localField, foreignField, as**.

- `from`: uma coleção no mesmo database para executar o `join`;
- `localField`: o campo da coleção de onde a operação de agregação está sendo executada. Será comparado por igualdade com o campo especificado no parâmetro `foreingField`;
- `foreingField`: o campo da coleção especificada no parâmetro `from` que será comparado com o campo `localField` por igualdade simples;
- `as`: o nome do novo *array* que será adicionado.

Mais 2 opcionais: **let** e **pipeline**

- `let`: define as variáveis que serão utilizadas no estágio *pipeline* dentro do `$lookup`. É necessário porque o estágio *pipeline* não consegue acessar diretamente os campos dos documentos de entrada, então esses campos precisam ser definidos previamente e transformados em variáveis;
- `pipeline`: define as condições ou o *pipeline* que será executado na coleção de junção. Se você quiser todos os documentos da coleção de junção, é só especificá-lo como vazio (`[]`).

```
db.orders.aggregate([
  {
    $lookup: {
      from: "warehouses",
      let: { order_item: "$item", order_qty: "$ordered" },
      pipeline: [
        {
          $match: {
            $expr: {
              $and: [
                { $eq: [ "$stock_item",  "$$order_item" ] },
                { $gte: [ "$instock", "$$order_qty" ] }
              ]
            }
          }
        },
        { $project: { stock_item: 0, _id: 0 } }
      ],
      as: "stockdata"
    }
  }
]);
```

## $group

**Faz agrupamentos**.
Principal parâmetro **_id** contendo campo(s) utilizado(s) para agrupamento.
Depois, operadores de acumulação, mais usados sendo **addToSet, avg, first, last, max, sum**.

- `$addToSet`: retorna um *array* com os valores únicos da expressão para cada grupo;
- `$avg`: retorna a média de valores numéricos. Valores não numéricos são ignorados;
- `$first`: retorna um valor do primeiro documento de cada grupo;
- `$last`: retorna um valor do último documento de cada grupo;
- `$max`: retorna o maior valor de cada grupo;
- `$sum`: retorna a soma de valores numéricos. Valores não numéricos são ignorados.

```
db.sales.aggregate([
  {
    $group: {
      _id: null,
      count: { $sum: 1 }
    }
  }
]);
// count sendo um nome aqui, como um alias
```


## $unwind

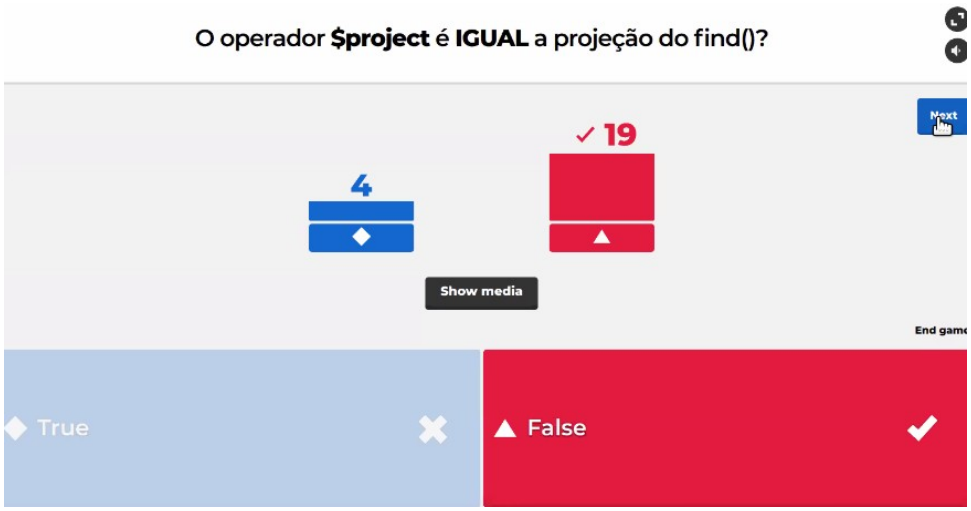
Trabalha com arrays. "Desconstrói" um campo array do documento de entrada e **gera como saída um documento para cada elemento do array**.

*db.collection.aggregate([{ $unwind : "$campo" }]);*

## $project

**Controla a exibição de campos.** Pode passar adiante no pipeline apenas alguns campos dos documentos vindos do estágio anterior, ou também criar novos via cálculo ou concatenação.

```
// Para incluir apenas oe campos title e author no documento de saída, exluindo _id
db.books.aggregate(
  [
    {
      $project : {
        _id: 0,
        title : 1,
        author : 1
      }
    }
  ]
);
```




Project faz mais, por exemplo:
Ajuda também para **renomear** adicionando *newName: "$formerName"*. (1 implicito, inclui).

<u>Dicas diversas</u>

Ideia de **Fordismo** no fluxo do Aggregation Framework: enquanto stages 2 e 3 trabalham, o 1 continua recebendo dado sem parar a produção.

Conceito de **pipe: a saída de um comando serve de entrada para o outro**.

Comparando com Updates no MongoDB com filtro e depois projection: aqui no AF o filtro match nem sempre é primeira etapa, boa pratica de performance mas pode não colar com logica procurada.

---

## 2) Aggregation Framework - Parte 2

Executar operações em um pipeline e adicionar novos campos aos documentos durante um pipeline.

### Expressão $add

Para **somar valores numéricos ou datas** (em milisegundos).

```
db.sales.aggregate([
  { $project: { item: 1, total: { $add: ["$price", "$fee"] } } }
]);
```

*// cria um novo campo com o valor total somando os campos price e fee*

```
db.sales.aggregate([
  { $project: { item: 1, billing_date: { $add: ["$date", 2.592e+8] } } }
]);
```

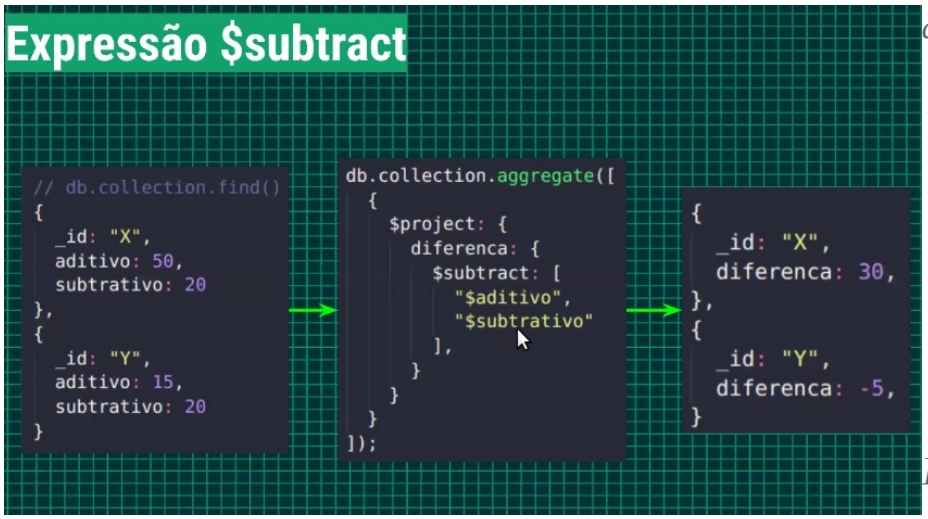
*// mesmo resultado:*
*// e+8 indica 8 casas depois de virgula*

```
db.sales.aggregate([
  { $project: { item: 1, billing_date: { $add: ["$date", 3 * 24 * 60 * 60000] } } }
]);
```

*// ou podemos adicionar a operação correspondendo com o tempo que queremos integrar*

### Expressão $subtract

Para **subtrair valores numéricos ou datas** (em milisegundos).




```
db.sales.aggregate([
  {
    $project: {
      item: 1,
      total: {
        $subtract: [
          { $add: ["$price", "$fee"] },
          "$discount"
        ]
      }
    }
  }
]);
```

*//$add para calcular o total e $subtract para aplicar um desconto no subtotal*

```
db.sales.aggregate([
  {
    $project: {
      item: 1,
      dateDifference: {
        $subtract: [new Date(), "$date"]
      }
    }
  }
]);
```

## Expressão $ceil

**Retorna o menor número inteiro maior ou igual ao número especificado** (arredonda para cima, teto).
*db.samples.aggregate([*
*{ $project: { value: 1, ceilingValue: { $ceil: "$value" } } }*
*]);*
*// também retorna valor original*

## Expressão $floor

**Retorna o maior número inteiro maior ou igual ao número especificado** (arredonda para baixo, chão).
*db.samples.aggregate([*
*{ $project: { value: 1, floorValue: { $floor: "$value" } } }*
*]);*

## Expressão $abs

**Retorna o valor absoluto** de um número, muito útil para **encontrar a diferença** entre dois valores.
*$abs: { $subtract: ["$start", "$end"] }*


## Expressão $multiply

**Multiplica** dois valores numéricos.
*total: {*
*$multiply: ["$price", "$quantity"]*
*}*

## Expressão $divide

**Divide** dois valores numéricos.
*workdays: {*
*$divide: ["$hours", 8]*
*}*

## Expressão $addFields

**Estágio que adiciona novos campos** aos documentos.
Um pipeline pode conter mais de um estágio $addFields.

*$addFields: {*
*totalHomework: { $sum: "$homework" } ,*
*totalQuiz: { $sum: "$quiz" }*
*}*

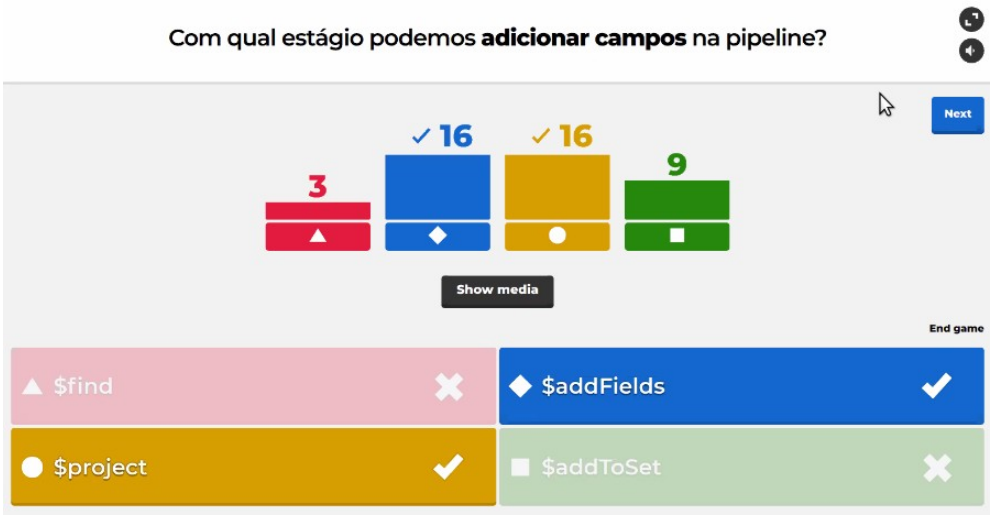
```
db.scores.aggregate([
  {
    $addFields: {
      totalHomework: { $sum: "$homework" } ,
      totalQuiz: { $sum: "$quiz" }
    }
  },
  {
    $addFields: {
      totalScore: {
        $add: [ "$totalHomework", "$totalQuiz", "$extraCredit" ]
      }
    }
  }
]);
```

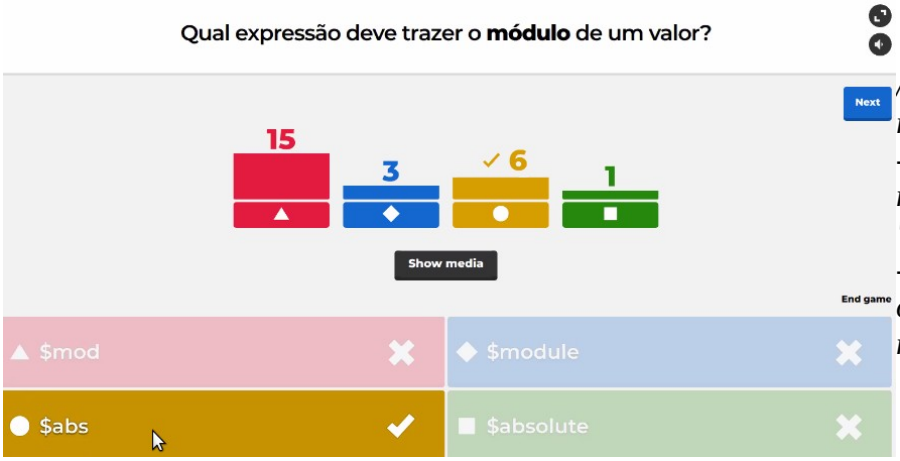
*// O primeiro estágio adiciona os campos totalHomework e totalQuiz, o segundo o campo totalScore, que soma os valores dos campos totalHomework, totalQuiz e extraCredit.*

<u>Dicas diversas</u>


*// subtract pode aceitar apenas 2 parâmetros, sendo que o segundo subtrai o primeiro*

*// como contornar:*
*$subtract: [ { $subtract: [ $valor1, $valor2 ] }, $valor3 ].*






*//Módulo ou Módulos pode referir-se a:*
*- Valor absoluto, uma função matemática, chamada ainda "módulo"*
*- Operação módulo, encontra o resto da divisão de um número por outro.*

*Diferença $sum e $add:* o primeiro é disponível na aggregation.

**Cursor:** Qualquer consulta. Tentar por exemplo .toArray() cria trabalho para MongoDB, que consume RAM. Solução sendo de primeiro armazenar dentro de variável, function.

```
const exemploFuncao = () => {
  const escopoPequeno = db.voos.find().limit(10).toArray();
  return escopoPequeno;
};
```

*Método* ***.itcount()*** para conferir o número de documentos retornados pelo pipeline.

Achar top 10 com sort e limit:

```
{
   $sort: {
     totalCompras: -1
   }
  },
  { $limit: 10 }
```

Desconto de 10% com subtract e multiply:

```
 {
   $addFields: {
     "compras.valorComDesconto": {
       $subtract: [
         "$compras.valorTotal",
         { $multiply: ["$compras.valorTotal", 0.10] }
       ]
     }
   }
```

Para selecionar e dar nome de status associado:

```
$match: {
     dataVenda: {
       $gte: ISODate('2020-03-01'),
       $lte: ISODate('2020-03-31')
     },
     status: "EM SEPARACAO"
```

Aprendizados do Projeto Aggregations

Achar item n° 25: skip 24 limit 1
Novos operadores usados: split, setIntersection, stdDevSamp , toInt, dayOfWeek .

Testes do projeto
ESLINT
TESTE LOCAL + RESTAURAR BD
./scripts/evaluate.sh && DBNAME=aggregations ./scripts/resetdb.sh assets

**Instalar MongoDB Compass**
- Download no link https://docs.mongodb.com/compass/master/install
- sudo apt update
- sudo apt install ./path ou seja: sudo apt install ./Documents/mongodb-compass_1.23.0_amd64.deb
- Criar Cluster para conectar no Compass (opcional)
- Entrar Compass: mongodb+srv://juliettebeaudet:<password>@clusterju.av0h3.mongodb.net/test