UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

TECNOLOGIAS DE PROGRAMAÇÃO DE SISTEMAS DE INFORMAÇÃO

# MongoDB

SISTEMAS GESTORES DE BASES DE DADOS II| Prof. Magno Andrade | Prof. Jorge Louro

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II


- **Conteúdo da apresentação - Parte 1**
  - Análise de dados
    - Agregação
- **Conteúdo da apresentação - Parte 2 (próxima aula)**
  - Análise de dados
    - MapReduce
    - Agregações únicas

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

# ANÁLISE DE DADOS

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Introdução**
  - As operações de agregação processam dados guardados e retornam valores que foram sujeitos a uma computação.
  - Os valores de múltiplos documentos são agrupados e são realizadas várias operações nesses dados de forma a retornar um único resultado.


- **Introdução**
  - Existem três formas de fazer agregação de dados:
    - Agregação
    - *MapReduce*
    - Agregações únicas

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

# ANÁLISE DE DADOS - AGREGAÇÃO

- **Agregação**
  - É uma *framework* para a agregação de dados, com o conceito de processamento de dados em *pipeline*.
  - A entrada são os documentos que passam por este *pipeline* de várias etapas e que transformam os documentos em resultados agregados.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação**
  - É uma alternativa ao *MapReduce*.
  - Deve ser a solução quando as funções a serem implementadas no *MapReduce* são muito complexas.

- **Agregação - *Pipeline***
  - O *pipeline* de agregação consiste em etapas.
  - Cada etapa transforma os documentos à medida que passam pelo *pipeline*.
  - O *pipeline* é executado com o comando *aggregate.*
  - O comando tem a seguinte sintaxe:
    - db.collection.aggregate( [ {$stage}, {$stage} ] )

- **Agregação - *Pipeline***
  - Não é necessário produzir um documento de saída para cada documento de entrada.
  - Algumas etapas podem gerar novos documentos ou filtrar.
  - As etapas podem aparecer várias vezes no *pipeline*.

- **Agregação - Expressões do *Pipeline***
  - As expressões podem incluir caminhos para os campos e variáveis de sistema, literais, objectos e operadores de expressão. As expressões podem ser “aninhadas”.
  - Para acessar aos campos dos documentos de entrada é necessário utilizar o caminho para o campo, da seguinte forma:
    - ***$****fieldname*
    - ***$****user.name* → para campos “aninhados” e que representa o campo ***user.name***

- **Agregação - Expressões do *Pipeline***
  - ***$<field>*** é equivalente a "***$$CURRENT.<field>***", em que ***CURRENT*** é a variável de sistema que por omissão representa a raiz do objecto actual nas etapas do *pipeline*.
  - ***Literals***
    - Usado para evitar que o uso do caracter ***$*** seja interpretado como um caminho para o campo, e que números/booleanos sejam interpretados como ***flags*** na etapa de projecção.

- **Agregação - Expressões do *Pipeline***
  - Objectos de expressão:
    - Tem a seguinte sintaxe:
      - { <field1>: <expression1>, ... }
    - Se as expressões são numéricas ou booleanas, estes são tratados como *flags* de projecção (ex: 1 ou ***true*** para incluir o campo), válido apenas na etapa ***$project***.
    - Para evitar isso utilizamos o operador ***$literal***.

- **Agregação - Expressões do *Pipeline***
  - Operador ***$literal***:
    - Retorna um valor sem analisar. Usar para valores em que o *pipeline* pode interpretar como uma expressão.
    - Tem a seguinte sintaxe:
      - { $literal: <value> }

- **Agregação - Expressões do *Pipeline***
  - Operador ***$literal***:
    - Funcionamento:
      - { $literal: { $add: [ 2, 3 ] } } **->** { "$add" : [ 2, 3 ] }
      - { $literal: { $literal: 1 } } **->** { "$literal" : 1 }
    - Se o <value> é uma expressão, o operador não avalia a expressão e retorna a expressão não avaliada.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Expressões do *Pipeline***
  - Operador ***$literal*** - Exemplo 1
    - Tendo em conta os seguintes documentos:
      - { "_id" : 1, "item" : "abc123", price: "$2.50" }
      - { "_id" : 2, "item" : "xyz123", price: "1" }
      - { "_id" : 3, "item" : "ijk123", price: "$1" }

- **Agregação - Expressões do *Pipeline***
  - Operador ***$literal*** - Exemplo 1
    - Pretendemos avaliar se o campo ***price*** é igual ao valor ***$1***, se utilizarmos a seguinte expressão (**errada**):
      - $eq: [ "$price", "$1" ]
      - Será realizada uma verificação de igualdade entre o valor do campo ***price*** e o valor do campo **1**.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Expressões do *Pipeline***
  - Operador ***$literal*** - Exemplo 1
    - Aplicando a seguinte etapa:
      - db.records.aggregate( [

        { $project: { costsOneDollar: { $eq: [ "$price", { $literal: "$1" } ] } } }

        ] )
    - Obtemos o seguinte resultado:
      - { "_id" : 1, "costsOneDollar" : false }
      - { "_id" : 2, "costsOneDollar" : false }
      - { "_id" : 3, "costsOneDollar" : true }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Alguns Operadores Aritméticos

| **Nome** | **Descrição** |
|---|---|
| $abs | Retorna o valor absoluto de um número. |
| $add | Adiciona números para retornar a soma deles. |
| $multiply | Multiplica os números e retorna o seu produto. |
| $subtract | Retorna o resultado da subtracção de dois valores. |
| $trunc | Retorna a parte inteira de um número. |

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores Aritméticos - ***$abs***
    - Retorna o valor absoluto de um número.
    - Tem a seguinte sintaxe:
      - { $abs: <number> }
    - **Nota**: <number> pode ser qualquer uma expressão válida desde que esta retorne um número.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores Aritméticos - ***$add***
    - Adição de números ou adição de números e uma data. Se um dos argumentos é uma data, o operador trata o outro argumento como milissegundos para adicionar à data.

    - Tem a seguinte sintaxe:
      - { $add: [ <expression1>, <expression2>, ... ] }

    - <expression1> pode ser uma expressão válida desde que esta retorne tudo números, ou números e uma data.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores Aritméticos - ***$add*** - Exemplo 1
    - Tendo em conta os seguintes documentos:
      - { "_id" : 1, "item" : "abc", "price" : 10, "fee" : 2, date: ISODate("2014-03-01T08:00:00Z") }
      - { "_id" : 2, "item" : "jkl", "price" : 20, "fee" : 1, date: ISODate("2014-03-01T09:00:00Z") }
      - { "_id" : 3, "item" : "xyz", "price" : 5,  "fee" : 0, date: ISODate("2014-03-15T09:00:00Z") }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores Aritméticos - ***$add*** - Exemplo 1
    - Aplicando a seguinte etapa:
      - db.sales.aggregate(
        [
        { $project: { item: 1, total: { $add: [ "$price", "$fee" ] } } }
        ]
        )
    - Obtemos o seguinte resultado:
      - { "_id" : 1, "item" : "abc", "total" : 12 }
      - { "_id" : 2, "item" : "jkl", "total" : 21 }
      - { "_id" : 3, "item" : "xyz", "total" : 5 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores Aritméticos - ***$trunc***
    - Obtém a parte inteira de um número.
    - Tem a seguinte sintaxe:
      - { $trunc: <number> }
    - O argumento <number> pode ser uma qualquer expressão válida desde que esta retorne um número.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores Aritméticos - ***$trunc*** - Exemplo
    - Tendo em conta a seguinte colecção:
      - { _id: 1, value: 9.25 }
      - { _id: 2, value: 8.73 }
      - { _id: 3, value: 4.32 }
      - { _id: 4, value: -5.34 }
    - Aplicando a seguinte etapa:
      - db.samples.aggregate([
        { $project: { value: 1, truncatedValue: { $trunc: "$value" } } }
        ])

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores Aritméticos - ***$trunc*** - Exemplo
    - Obtemos os seguintes resultados:
      - { "_id" : 1, "value" : 9.25, "truncatedValue" : 9 }
      - { "_id" : 2, "value" : 8.73, "truncatedValue" : 8 }
      - { "_id" : 3, "value" : 4.32, "truncatedValue" : 4 }
      - { "_id" : 4, "value" : -5.34, "truncatedValue" : -5 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

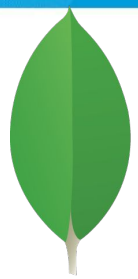
- **Agregação - Operadores de expressão**
  - Alguns Operadores de Data

| **Nome** | **Descrição** |
|---|---|
| $year | Retorna o ano de uma data. (número) |
| $month | Retorna o mês de uma data. (número) |
| $week | Retorna o número da semana de uma data. (número) |
| $hour | Retorna a hora de uma data. (número) |
| $minute | Retorna o minuto de uma data. (número) |
| $second | Retorna os segundos de uma data. (número) |

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II


- **Agregação - Operadores de expressão**
  - Operadores de Data - ***$year***
    - Retorna em número (ex: 2014) o ano de uma data.
    - Tem a seguinte sintaxe:
      - { $year: <dateExpression> }
    - **Nota**: este operador **não** aceita o argumento da data em *string*. Apenas um campo do tipo data, Timestamp ou ObjectID.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II



mongoDB®

- **Agregação - Operadores de expressão**
  - Operadores de Data - ***$year***
    - Funcionamento - Exemplos

| **Exemplo** | **Resultado** |
|---|---|
| { $year: new Date("2016-01-01") } | 2016 |
| { $year: { date: new Date("Jan 7, 2003") } } | 2003 |
| { $year: {<br>date: new Date("August 14, 2011"),<br>timezone: "America/Chicago"<br>} } | |
| { $year: ISODate("1998-11-07T00:00:00Z") } | 1998 |

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II


- **Agregação - Operadores de expressão**
  - Operadores de Data - ***$year*** - Exemplo
    - **Nota**: O exemplo seguinte aplica-se em termos de sintaxe aos restantes operadores de data, mudando apenas o operador na sintaxe.
    - Tendo em conta o seguinte documento:
      - {
        "_id" : 1,
        "item" : "abc",
        "price" : 10,
        "quantity" : 2,
        "date" : ISODate("2014-01-01T08:15:39.736Z")
        }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores de Data - ***$year*** - Exemplo
    - Aplicando a seguinte etapa:


```
db.sales.aggregate(
  [
    {
      $project:
        {
          year: { $year: "$date" },
          month: { $month: "$date" },
          day: { $dayOfMonth: "$date" },
          hour: { $hour: "$date" },
          minutes: { $minute: "$date" },
          seconds: { $second: "$date" },
          milliseconds: { $millisecond: "$date" },
          dayOfYear: { $dayOfYear: "$date" },
          dayOfWeek: { $dayOfWeek: "$date" },
          week: { $week: "$date" }
        }
    }
  ]
)
```

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores de Data - ***$year*** - Exemplo
    - Obtemos o seguinte resultado:

```
{
  "_id" : 1,
  "year" : 2014,
  "month" : 1,
  "day" : 1,
  "hour" : 8,
  "minutes" : 15,
  "seconds" : 39,
  "milliseconds" : 736,
  "dayOfYear" : 1,
  "dayOfWeek" : 4,
  "week" : 0
}
```

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II


- **Agregação - Operadores de expressão**
  - Alguns Operadores de *Arrays*

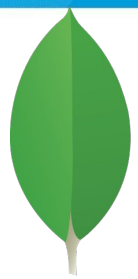
| **Nome** | **Descrição** |
|---|---|
| $arrayElemAt | Retorna o elemento do *array* especificado pelo índice. |
| $concatArrays | Concatena *arrays* para retornar um só *array* concatenado. |
| $filter | Selecciona um subconjunto do *array* para retornar um *array* com apenas elementos que satisfazem a condição de filtro. |
| $in | Retorna um booleano a indicar se o valor especificado está no *array*. |
| $reverseArray | Retorna um *array* com os elementos em ordem inversa. |
| $slice | Retorna um subconjunto de um *array*. |

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Operadores de expressão**
  - Operadores Booleanos

| **Nome** | **Descrição** |
|---|---|
| $and | Retorna ***true*** apenas quando todas as expressões são avaliadas a verdadeiro. Aceita qualquer número de argumentos nas expressões. |
| $not | Retorna um valor booleano que é o oposto da expressão de argumento. Aceita uma única expressão de argumento. |
| $or | Retorna ***true*** quando alguma das expressões é avaliada a verdadeiro. Aceita qualquer número de argumentos nas expressões. |



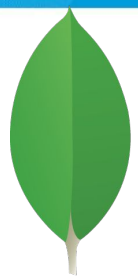
mongoDB®

- **Agregação - Operadores de expressão**
  - Operadores Booleanos - ***$and*** - Exemplo
    - Tendo em conta os seguintes documentos:
      - { "_id" : 1, "item" : "abc1", description: "product 1", qty: 300 }
      - { "_id" : 2, "item" : "abc2", description: "product 2", qty: 200 }
      - { "_id" : 3, "item" : "xyz1", description: "product 3", qty: 250 }
      - { "_id" : 4, "item" : "VWZ1", description: "product 4", qty: 300 }
      - { "_id" : 5, "item" : "VWZ2", description: "product 5", qty: 180 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores Booleanos - ***$and*** - Exemplo
    - Aplicando a seguinte etapa:

```
db.inventory.aggregate(
   [
     {
       $project:
          {
            item: 1,
            qty: 1,
            result: { $and: [ { $gt: [ "$qty", 100 ] }, { $lt: [ "$qty", 250 ] } ] }
          }
     }
   ]
)
```



mongoDB®

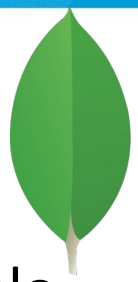
- **Agregação - Operadores de expressão**
  - Operadores Booleanos - ***$and*** - Exemplo
    - Obtemos o seguinte resultado:
      - { "_id" : 1, "item" : "abc1", "result" : false }
      - { "_id" : 2, "item" : "abc2", "result" : true }
      - { "_id" : 3, "item" : "xyz1", "result" : false }
      - { "_id" : 4, "item" : "VWZ1", "result" : false }
      - { "_id" : 5, "item" : "VWZ2", "result" : true }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores de Comparação

| **Nome** | **Descrição** |
|---|---|
| $cmp | Retorna 0 se dois valores são equivalentes, 1 se o primeiro valor é maior que o segundo, -1 se o primeiro valor é menor que o segundo. |
| $eq | Retorna ***true*** se os valores são equivalentes. |
| $gt | Retorna ***true*** se o primeiro valor é maior que o segundo. |
| $gte | Retorna ***true*** se o primeiro valor é maior ou igual que o segundo. |
| $lt | Retorna ***true*** se o primeiro valor é menor que o segundo. |
| $lte | Retorna ***true*** se o primeiro valor é menor ou igual ao segundo. |
| $ne | Retorna ***true*** se os valores não são equivalentes. |

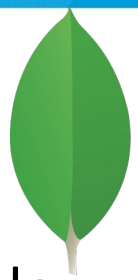
UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II



mongoDB®

- **Agregação - Operadores de expressão**
  - Operadores de Comparação - ***$eq*** - Exemplo
    - Compara dois valores, retorna ***true*** se os valores forem equivalentes, caso contrário ***false***.
    - Tem a seguinte sintaxe:
      - { $eq: [ <expression1>, <expression2> ] }
    - Este operador compara o valor como também o tipo de dados.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II



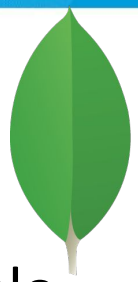
mongoDB®

- **Agregação - Operadores de expressão**
  - Operadores de Comparação - ***$eq*** - Exemplo
    - Tendo em conta os seguintes documentos:
      - { "_id" : 1, "item" : "abc1", description: "product 1", qty: 300 }
      - { "_id" : 2, "item" : "abc2", description: "product 2", qty: 200 }
      - { "_id" : 3, "item" : "xyz1", description: "product 3", qty: 250 }
      - { "_id" : 4, "item" : "VWZ1", description: "product 4", qty: 300 }
      - { "_id" : 5, "item" : "VWZ2", description: "product 5", qty: 180 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores de Comparação - ***$eq*** - Exemplo
    - Aplicando a seguinte etapa:

```
db.inventory.aggregate(
   [
     {
       $project:
          {
            item: 1,
            qty: 1,
            qtyEq250: { $eq: [ "$qty", 250 ] },
            _id: 0
          }
     }
   ]
)
```

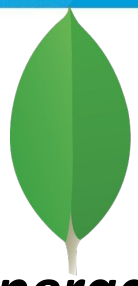


mongoDB®

- **Agregação - Operadores de expressão**
  - Operadores de Comparação - ***$eq*** - Exemplo
    - É comparado se o campo ***qty*** é equivalente a 250, em cada um dos documentos.

    - Obtemos o seguinte resultado:
      - { "item" : "abc1", "qty" : 300, "qtyEq250" : false }
      - { "item" : "abc2", "qty" : 200, "qtyEq250" : false }
      - { "item" : "xyz1", "qty" : 250, "qtyEq250" : true }
      - { "item" : "VWZ1", "qty" : 300, "qtyEq250" : false }
      - { "item" : "VWZ2", "qty" : 180, "qtyEq250" : false }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II



mongoDB®

- **Agregação - Operadores de expressão**
  - Operadores de Comparação - ***Restantes operadores***
    - A aplicação destes operadores em termos de sintaxe é igual aos exemplos nos slides anteriores (ex: ***$eq***).

    - Estes operadores comparam o valor como também o tipo de dados.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Alguns Operadores de Condição

| **Nome** | **Descrição** |
|---|---|
| $cond | Um operador ternário que avalia uma expressão e dependendo do resultado retorna o valor de uma das outras duas expressões. Aceita três expressões numa lista ordenada ou três parâmetros nomeados. |
| $ifNull | Retorna o resultado não nulo da primeira expressão ou o resultado da segunda expressão, se a primeira expressão resultar em um resultado nulo. Este resultado ***null*** abrange as instâncias de valores não definidos ou campos inexistentes. Aceita duas expressões como argumentos. O resultado da segunda expressão pode ser ***null***. |

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II


- **Agregação - Operadores de expressão**
  - Operadores de Condição - ***$cond***
    - Avalia uma expressão booleana e retorna uma de duas expressões especificadas.
    - Tem duas sintaxes:
      - { $cond: { if: <boolean-expression>, then: <true-case>, else: <false-case-> } }
      - ou
      - { $cond: [ <boolean-expression>, <true-case>, <false-case> ] }
    - Os argumentos podem ser qualquer expressão válida.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores de Condição - ***$cond***
    - Funcionamento
      - Se a <boolean-expression> for verdadeira, então o operador retorna o valor da expressão <true-case>. Caso contrário, o operador retorna o valor da expressão <false-case>.

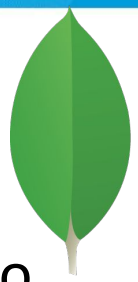
    - Exemplo - Tendo em conta os seguintes documentos:
      - { "_id" : 1, "item" : "abc1", qty: 300 }
      - { "_id" : 2, "item" : "abc2", qty: 200 }
      - { "_id" : 3, "item" : "xyz1", qty: 250 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Operadores de expressão**
  - Operadores de Condição - ***$cond*** - Exemplo
    - Aplicando a seguinte etapa:
      - db.inventory.aggregate( [

```
{
  $project:
   {
     item: 1,
     discount:
      {
        $cond: { if: { $gte: [ "$qty", 250 ] }, then: 30, else: 20 }
      }
   }
} ] )
```

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II



mongoDB®

- **Agregação - Operadores de expressão**
  - Operadores de Condição - ***$cond*** - Exemplo
    - A etapa seguinte é utilizada para colocar o valor de desconto a 30 se o campo ***qty*** é maior ou igual a 250 e a 20 se o campo ***qty*** é menor que 250.

    - Obtemos os seguintes resultados:
      - { "_id" : 1, "item" : "abc1", "discount" : 30 }
      - { "_id" : 2, "item" : "abc2", "discount" : 20 }
      - { "_id" : 3, "item" : "xyz1", "discount" : 30 }


- **Agregação - Funcionamento do *Pipeline***
  - O comando *aggregate* é realizado em apenas uma única colecção.
  - Todos os documentos da colecção são passados para o *pipeline*.
  - Para melhorar o funcionamento é recomendado efectuar as seguintes optimizações.

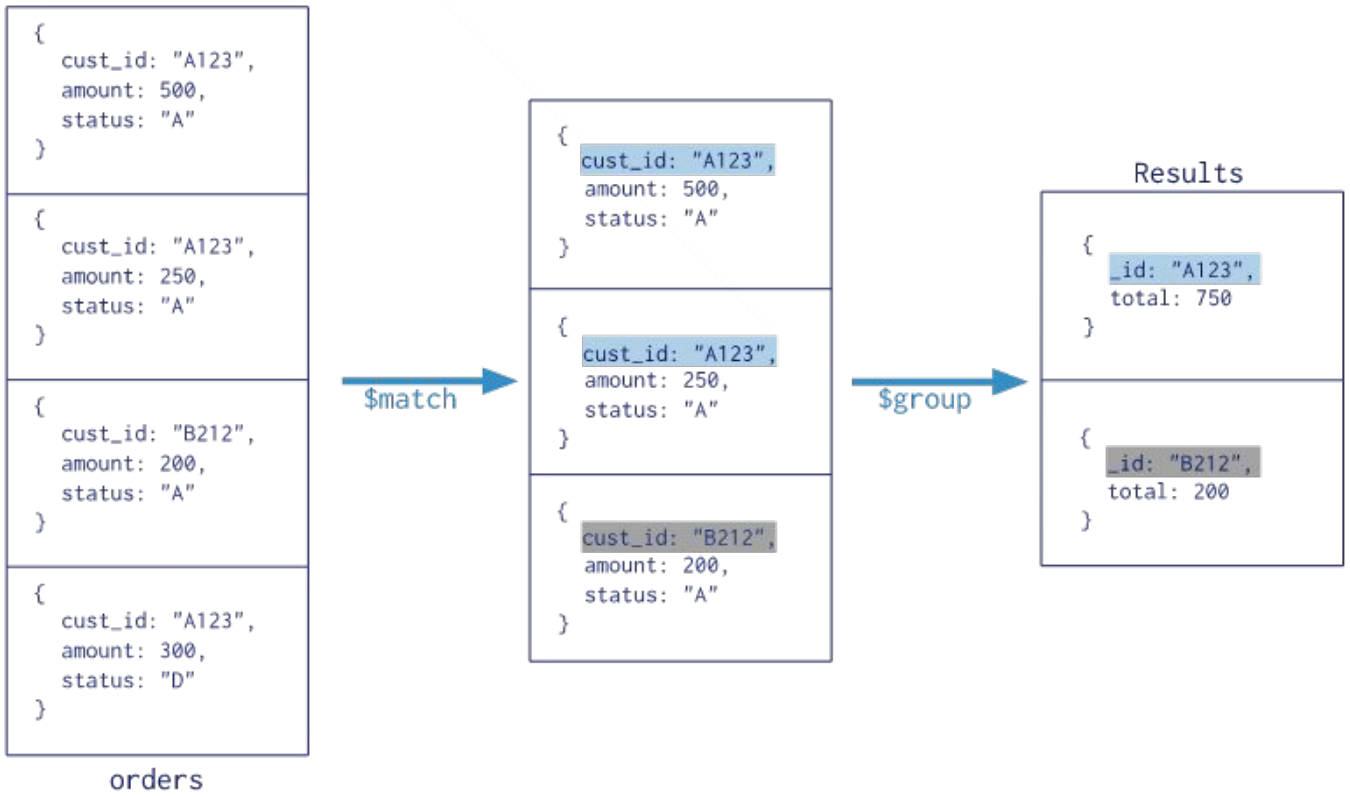

- **Agregação - Funcionamento do *Pipeline***
  - **Optimizações**
    - Operadores de *pipeline* e índices
      - Os operadores ***$match*** e ***$sort*** podem tirar vantagens dos índices quando colocados no início do *pipeline*.
      - Se a operação de agregação apenas necessita de uma amostra da colecção é recomendado utilizar as etapas, ***$match, $limit, e $skip*** no início do pipeline.
      - Sempre que possível, devemos colocar estas etapas sempre no início.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB.

- **Exemplo**

```
Collection
db.orders.aggregate( [
    $match stage ──►   { $match: { status: "A" } },
    $group stage ──►   { $group: { _id: "$cust_id",total: { $sum: "$amount" } } }
                     ] )
```



UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação**


- **Agregação - Etapas mais comuns do *Pipeline***
  - ***$match*** -> filtra o número de documentos que é passado para a próxima etapa.
  - ***$group*** -> agrupa os documentos por uma chave distinta, esta chave também pode ser composta.
  - ***$project*** -> passa os documentos com os campos especificados ou novos campos computados para a próxima etapa.


- **Agregação - Etapas mais comuns do *Pipeline***
  - ***$sort*** -> retorna os documentos de entrada ordenados.
  - ***$limit*** -> limita o número de documentos para próxima etapa.
  - ***$unwind*** -> divide um *array*, em que cada elemento deste, é um documento.
  - ***$out*** -> é a última etapa, escreve/modifica os documentos resultantes numa colecção.


- **Agregação - Etapa *$match***
  - Passa para a próxima etapa, apenas os documentos que satisfazem um determinado critério.
  - É recomendado ser a primeira etapa do *pipeline*.
  - Se colocado como primeira etapa, tira partido dos índices.


- **Agregação - Etapa *$match***
  - Tem o seguinte formato:
    - { $match: { <query> } }
    - <query> recebe um critério ou um conjunto de critérios que têm de se satisfeitos.
  - O critério de pesquisa é idêntico ao utilizado nas operações de ***Read***.
  - Os operadores de comparação e lógicos podem ser utilizados.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Etapa *$match***
  - **Exemplo 1**
    - Tendo em conta a seguinte colecção:

```
_id" : ObjectId("512bc95fe835e68f199c8686"), "author" : "dave", "score" : 80, "views" : 100
_id" : ObjectId("512bc962e835e68f199c8687"), "author" : "dave", "score" : 85, "views" : 521
_id" : ObjectId("55f5a192d4bede9ac365b257"), "author" : "ahn", "score" : 60, "views" : 1000
_id" : ObjectId("55f5a192d4bede9ac365b258"), "author" : "li", "score" : 55, "views" : 5000 }
_id" : ObjectId("55f5a1d3d4bede9ac365b259"), "author" : "annT", "score" : 60, "views" : 50 }
_id" : ObjectId("55f5a1d3d4bede9ac365b25a"), "author" : "li", "score" : 94, "views" : 999 }
_id" : ObjectId("55f5a1d3d4bede9ac365b25b"), "author" : "ty", "score" : 95, "views" : 1000 }
```

mongoDB®

- **Agregação - Etapa *$match***
  - **Exemplo 1**
    - Aplicando a etapa nesta colecção:
      - db.articles.aggregate(

        [ { $match : { author : "dave" } } ]

        );
    - Obtemos todos os documentos em que o campo ***author*** é igual a "**dave**":

```
{ "_id" : ObjectId("512bc95fe835e68f199c8686"), "author" : "dave", "score" : 80,
{ "_id" : ObjectId("512bc962e835e68f199c8687"), "author" : "dave", "score" : 85,
```


- **Agregação - Etapa *$match***
  - **Exemplo 2**
    - Aplicando a etapa nesta colecção:
      - db.articles.aggregate( [

        { $match: { $or: [ { score: { $gt: 70, $lt: 90 } }, { views: { $gte: 1000 } } ] } },

        { $group: { _id: null, count: { $sum: 1 } } }

        ] );


- **Agregação - Etapa *$match***
  - **Exemplo 2**
    - Na etapa ***$match*** são obtidos todos os documentos em que o campo ***score*** é maior que 70 **e** menor que 90 **ou** o campo ***views*** é maior ou igual a 1000.
    - Na etapa ***$group*** são contados todos os documentos obtidos.
    - Obtemos:

```
{ "_id" : null, "count" : 5 }
```


- **Agregação - Etapa *$group***
  - Agrupa todos os documentos de entrada especificado por um expressão identificadora.
  - Aplica as expressões de acumulação, se especificadas, a cada grupo.

- **Agregação - Etapa *$group***
  - Realiza a saída de um documento, por cada grupo distinto.
  - Os documentos de saída apenas tem o campo identificador, e se especificado, os campos acumuladores.
  - **Nota**: não ordena os documentos de saída.


- **Agregação - Etapa *$group***
  - **Tem o seguinte formato**
    - { $group: { _id: <expression>, <field1>: { <accumulator1> : <expression1> }, ... } }
      - O campo ***_id*** é obrigatório.
      - O campo ***_id*** pode ser ***null,*** para calcular os valores acumulados para todos os documentos de entrada como um todo.
      - Os restantes campos computados são opcionais.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II


- **Agregação - Etapa *$group***
  - Alguns Operadores de Acumulação

| **Nome** | **Descrição** |
|---|---|
| $avg | Retorna uma média dos valores numéricos. Ignora valores não numéricos. |
| $first | Retorna um valor do primeiro documento de cada grupo. A ordem só é definida se os documentos estão numa ordem definida. |
| $last | Retorna um valor do último documento de cada grupo. A ordem só é definida se os documentos estão numa ordem definida. |
| $max | Retorna o maior valor de cada grupo. |


- **Agregação - Etapa *$group***
  - Alguns Operadores de acumulação

| **Nome** | **Descrição** |
|---|---|
| $min | Retorna o menor valor de cada grupo. |
| $push | Retorna um *array* de valores de expressão para cada grupo. |
| $addToSet | Retorna um *array* de valores únicos de expressão para cada grupo. A ordem dos elementos do *array* é indefinida. |
| $sum | Retorna a soma de valores numéricos. Ignora os valores não numéricos. |

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$group***
  - Operadores de Acumulação - ***$avg***
    - Retorna uma média dos valores numéricos. Ignora valores não numéricos.
    - Tem a seguinte sintaxe:
      - { $avg: <expression> }
    - Retorna a média total de todos os valores numéricos resultantes da aplicação de uma expressão especificada a cada documento num grupo de documentos que têm a mesma chave de grupo.

- **Agregação - Etapa *$group***
  - Operadores de Acumulação - ***$avg*** - Exemplo
    - Tendo em conta os documentos seguintes:
      - { "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-01-01T08:00:00Z") }
      - { "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-02-03T09:00:00Z") }
      - { "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 5, "date" : ISODate("2014-02-03T09:05:00Z") }
      - { "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-02-15T08:00:00Z") }
      - { "_id" : 5, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-02-15T09:12:00Z") }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$group***
  - Operadores de Acumulação - ***$avg*** - Exemplo
    - Aplicando a seguinte etapa:
      - db.sales.aggregate( [ {
        $group:
        {
        _id: "$item",
        avgAmount: { $avg: { $multiply: [ "$price", "$quantity" ] } },
        avgQuantity: { $avg: "$quantity" }
        }
        } ] )

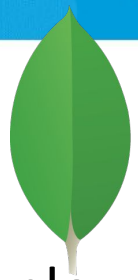
- **Agregação - Etapa *$group***
  - Operadores de Acumulação - ***$avg*** - Exemplo
    - Agrupamos os documentos pelo campo ***item*** e usamos o acumulador duas vezes para obter as médias para cada grupo apresentadas nos campos:
      - ***avgAmount*** e ***avgQuantity***
    - Obtemos os seguintes resultados:
      - { "_id" : "xyz", "avgAmount" : 37.5, "avgQuantity" : 7.5 }
      - { "_id" : "jkl", "avgAmount" : 20, "avgQuantity" : 1 }
      - { "_id" : "abc", "avgAmount" : 60, "avgQuantity" : 6 }

- **Agregação - Etapa *$group***
  - Operadores de Acumulação - ***$sum***
    - Calcula e retorna a soma de valores numéricos. Ignora valores não numéricos.

    - Retorna a soma colectiva de todos os valores numéricos, que resultam da aplicação de uma expressão específica em cada documento de um grupo de documentos, que partilham a mesma chave de grupo.

    - Tem a seguinte sintaxe:
      - { $sum: <expression> }

- **Agregação - Etapa *$group***
  - Operadores de Acumulação - ***$sum*** - Exemplo
    - Tendo em conta os seguintes documentos:
      - { "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-01-01T08:00:00Z") }
      - { "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-02-03T09:00:00Z") }
      - { "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 5, "date" : ISODate("2014-02-03T09:05:00Z") }
      - { "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-02-15T08:00:00Z") }
      - { "_id" : 5, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-02-15T09:05:00Z") }



mongoDB®

- **Agregação - Etapa *$group***
  - Operadores de Acumulação - ***$sum*** - Exemplo
    - Aplicando a seguinte etapa:
      - db.sales.aggregate( [

```
{
 $group:
  {
   _id: { day: { $dayOfYear: "$date"}, year: { $year: "$date" } },
   totalAmount: { $sum: { $multiply: [ "$price", "$quantity" ] } },
   count: { $sum: 1 }
  }
}
] )
```

- **Agregação - Etapa *$group***
  - Operadores de Acumulação - ***$sum*** - Exemplo
    - Agrupamos os documentos pelo dia (***day***) e ano (***year***) do campo ***date***, o acumulador para computar o valor total (***totalAmount***) e a contagem (***count***) para cada grupo de documentos.

    - Obtemos o seguintes resultados:
      - { "_id" : { "day" : 46, "year" : 2014 }, "totalAmount" : 150, "count" : 2 }
      - { "_id" : { "day" : 34, "year" : 2014 }, "totalAmount" : 45, "count" : 2 }
      - { "_id" : { "day" : 1, "year" : 2014 }, "totalAmount" : 20, "count" : 1 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Etapa *$group***
  - **Exemplo 1**
    - Tendo em conta a seguinte colecção:

```
"_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-03-01T08:00:
"_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-03-01T09:00:
"_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-03-15T09:00:
"_id" : 4, "item" : "xyz", "price" : 5, "quantity" : 20, "date" : ISODate("2014-04-04T11:21:
"_id" : 5, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-04-04T21:23
```

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$group***
  - **Exemplo 1**
    - Aplicando a etapa nesta colecção:
      - db.sales.aggregate(
        [ {  $group : {
        _id : { month: { $month: "$date" }, day: { $dayOfMonth: "$date" }, year: { $year: "$date" } } },
        totalPrice: { $sum: { $multiply: [ "$price", "$quantity" ] } },
        averageQuantity: { $avg: "$quantity" },
        count: { $sum: 1 }
        } } ] )

- **Agregação - Etapa *$group***
  - **Exemplo 1**
    - Agrupamos todos os documentos pelo mês, dia e ano e calculando o preço total e quantidade média, como também o número de documentos por cada grupo.
    - Obtemos o seguinte resultado:
      - { "_id" : { "month" : 3, "day" : 15, "year" : 2014 }, "totalPrice" : 50, "averageQuantity" : 10, "count" : 1 }
      - { "_id" : { "month" : 4, "day" : 4, "year" : 2014 }, "totalPrice" : 200, "averageQuantity" : 15, "count" : 2 }
      - { "_id" : { "month" : 3, "day" : 1, "year" : 2014 }, "totalPrice" : 40, "averageQuantity" : 1.5, "count" : 2 }


- **Agregação - Etapa *$group***
  - **Exemplo 2**
    - Aplicando a etapa nesta colecção:
      - db.sales.aggregate(

```
[ {  $group : {
      _id : null,
      totalPrice: { $sum: { $multiply: [ "$price", "$quantity" ] } },
      averageQuantity: { $avg: "$quantity" },
      count: { $sum: 1 }
    } } ] )
```

- **Agregação - Etapa *$group***
  - **Exemplo 2**
    - É calculado o preço total e quantidade média, como também o número de documentos de **todos** os documentos da colecção.
    - Obtemos o seguinte resultado:
      - { "_id" : null, "totalPrice" : 290, "averageQuantity" : 8.6, "count" : 5 }

mongoDB®

- **Agregação - Etapa *$group***
  - **Exemplo 3**
    - Tendo em conta a seguinte colecção:

```
{ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date"
{ "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date"
{ "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 10, "date"
{ "_id" : 4, "item" : "xyz", "price" : 5, "quantity" : 20, "date"
{ "_id" : 5, "item" : "abc", "price" : 10, "quantity" : 10, "date"
```

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$group***
  - **Exemplo 3**
    - Aplicando a etapa nesta colecção:
      - db.sales.aggregate( [ { $group : { _id : "$item" } } ] )
    - Agrupamos todos os elementos pelo ***item*** para obter todos os valores distintos.
    - Obtemos o seguinte resultado:
      - { "_id" : "xyz" }
      - { "_id" : "jkl" }
      - { "_id" : "abc" }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Etapa *$group***
  - **Exemplo 4**
    - Tendo em conta a seguinte colecção:

```
{ "_id" : 8751, "title" : "The Banquet", "author" : "Dante", "copies" : 2 }
{ "_id" : 8752, "title" : "Divine Comedy", "author" : "Dante", "copies" : 1 }
{ "_id" : 8645, "title" : "Eclogues", "author" : "Dante", "copies" : 2 }
{ "_id" : 7000, "title" : "The Odyssey", "author" : "Homer", "copies" : 10 }
{ "_id" : 7020, "title" : "Iliad", "author" : "Homer", "copies" : 10 }
```

- **Agregação - Etapa *$group***
  - **Exemplo 4**
    - Aplicando a etapa nesta colecção:
      - db.books.aggregate( [

        { $group : { _id : "$author", books: { $push: "$title" } } }

        ])
    - Agrupamos os títulos dos livros pelos autores.
    - Obtemos o seguinte resultado:
      - { "_id" : "Homer", "books" : [ "The Odyssey", "Iliad" ] }
      - { "_id" : "Dante", "books" : [ "The Banquet", "Divine Comedy", "Eclogues" ] }

- **Agregação - Etapa *$group***
  - **Exemplo 5**
    - Aplicando a etapa nesta colecção:
      - db.books.aggregate( [

        { $group : { _id : "$author", books: { $push: "$$ROOT" } } }

        ] )
    - Agrupamos os documentos por autor, usando a variável de sistema *$$ROOT*.
      - *$$ROOT* → referencia o documento raiz, ex: o documento no nível superior e que está actualmente a ser processado no *pipeline*.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$group***
  - **Exemplo 5**
    - Obtemos o seguinte resultado:

```
{
  "_id" : "Homer",
  "books" :
    [
      { "_id" : 7000, "title" : "The Odyssey", "author" : "Homer", "copies" : 10 },
      { "_id" : 7020, "title" : "Iliad", "aut{
    ]                                          "_id" : "Dante",
}                                              "books" :
                                                 [
                                                   { "_id" : 8751, "title" : "The Banquet", "author" : "Dante", "copies" : 2 },
                                                   { "_id" : 8752, "title" : "Divine Comedy", "author" : "Dante", "copies" : 1 },
                                                   { "_id" : 8645, "title" : "Eclogues", "author" : "Dante", "copies" : 2 }
                                                 ]
                                           }
```


- **Agregação - Etapa *$project***
  - Redesenha cada documento no *pipeline* com a adição de novos ou removendo campos existentes.
  - Para cada documento de entrada, existe um documento de saída.


- **Agregação - Etapa *$project***
  - **Restrições**
    - Se as <specification(s)> for um documento vazio, é produzido um erro.
  - **Tem o seguinte formato**
    - { $project: { <specification(s)> } }


- **Agregação - Etapa *$project***
  - Algumas notas:
    - O campo ***_id*** está por omissão incluído, para excluí-lo, é preciso fazê-lo explicitamente.
    - Para incluir outros campos é necessário especificá-los.


- **Agregação - Etapa *$project***
  - Algumas notas:
    - Se for especificado a inclusão de um campo que não existe, a etapa ignora este campo.
    - Para colocar o valor de um campo directamente num número ou booleano, é necessário utilizar o operador ***$literal***, porque a etapa trata o número ou booleano como uma "*flag*" para incluir ou excluir o campo.
      - Ex: db.collection.aggregate( [ { $project: { cost:  { $literal: 10 }  } }] )


- **Agregação - Etapa *$project***
  - Algumas notas:
    - Nos campos de documentos embebidos podem ser usadas duas notações:
      - "contact.address.country": <1 or 0 or expression>
      - ou
      - contact: { address: { country: <1 or 0 or expression> } }
    - Nos campos de documentos embebidos é inválida a seguinte notação:
      - contact: { "address.country": <1 or 0 or expression> }

mongoDB®

- **Agregação - Etapa *$project***
  - <specification(s)> tem as seguintes formas:

| **Forma** | **Descrição** |
|---|---|
| <field>: <1 or true> | Especifica a inclusão de um campo. |
| _id: <0 or false> | Especifica a exclusão do campo ***_id***. |
| <field>: <expression> | Adiciona um novo campo ou altera o valor do campo existente. |
| <field>:<0 or false> | Especifica a exclusão de um campo. |


- **Agregação - Etapa *$project***
  - Alguns Operadores de Acumulação

| **Nome** | **Descrição** |
|---|---|
| $avg | Retorna uma média dos valores numéricos da expressão especificada ou lista de expressões para cada documento. Ignora valores não numéricos. |
| $max | Retorna o maior valor da expressão especificada ou lista de expressões para cada documento. |
| $min | Retorna o menor valor da expressão especificada ou lista de expressões para cada documento. |
| $sum | Retorna a soma de valores numéricos. Ignora valores não numéricos. |

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$project***
  - Operador de acumulação - ***$avg***
    - Retorna a média de valor para todos os valores numéricos. Ignora os valores não numéricos.
    - Retorna a média da expressão especificada ou lista de expressões, para cada documento e tem duas sintaxes:
      - { $avg: <expression> } → uma só expressão
      - { $avg: [ <expression1>, <expression2> ... ] } → uma lista de expressões

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$project***
  - Operador de acumulação - ***$avg*** - Exemplo
    - Tendo em conta os seguintes documentos:
      - { "_id": 1, "quizzes": [ 10, 6, 7 ], "labs": [ 5, 8 ], "final": 80, "midterm": 75 }
      - { "_id": 2, "quizzes": [ 9, 10 ], "labs": [ 8, 8 ], "final": 95, "midterm": 80 }
      - { "_id": 3, "quizzes": [ 4, 5, 5 ], "labs": [ 6, 5 ], "final": 78, "midterm": 70 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$project***
  - Operador de acumulação - ***$avg*** - Exemplo
    - Aplicando a seguinte etapa:

```
db.students.aggregate([
  {
   $project: {
    quizAvg: { $avg: "$quizzes"},
    labAvg: { $avg: "$labs" },
    examAvg: { $avg: [ "$final", "$midterm" ] }
   }
  }
])
```

- **Agregação - Etapa *$project***
  - Operador de acumulação - ***$avg*** - Exemplo
    - É obtida a média das pontuações de ***quizzes***, pontuações de ***lab*** e a média entre o campo ***final*** e ***midterm***.

    - Obtemos os seguintes resultados:
      - { "_id" : 1, "quizAvg" : 7.666666666666667, "labAvg" : 6.5, "examAvg" : 77.5 }
      - { "_id" : 2, "quizAvg" : 9.5, "labAvg" : 8, "examAvg" : 87.5 }
      - { "_id" : 3, "quizAvg" : 4.666666666666667, "labAvg" : 5.5, "examAvg" : 74 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$project***
  - Operador de acumulação - ***$sum***
    - Calcula e retorna a soma de valores numéricos. Ignora valores não numéricos.

    - Retorna a soma da expressão específica ou lista de expressões, para cada documento e tem duas sintaxes:
      - { $sum: <expression> } → uma só expressão
      - { $sum: [ <expression1>, <expression2> ... ] } → uma lista de expressões

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$project***
  - Operador de acumulação - ***$sum*** - Exemplo
    - Tendo em conta os seguintes documentos:
      - { "_id": 1, "quizzes": [ 10, 6, 7 ], "labs": [ 5, 8 ], "final": 80, "midterm": 75 }
      - { "_id": 2, "quizzes": [ 9, 10 ], "labs": [ 8, 8 ], "final": 95, "midterm": 80 }
      - { "_id": 3, "quizzes": [ 4, 5, 5 ], "labs": [ 6, 5 ], "final": 78, "midterm": 70 }
    - Aplicando a seguinte etapa:
      - db.students.aggregate([ {

```
$project: {
  quizTotal: { $sum: "$quizzes"},
  labTotal: { $sum: "$labs" },
  examTotal: { $sum: [ "$final", "$midterm" ] }
} } ])
```

- **Agregação - Etapa *$project***
  - Operador de acumulação - ***$sum*** - Exemplo
    - Assim, obtemos a soma total das pontuações ***quiz***, ***lab*** e o total entre o campo ***final*** e ***midterm***.

    - Obtemos os seguintes resultados:
      - { "_id" : 1, "quizTotal" : 23, "labTotal" : 13, "examTotal" : 155 }
      - { "_id" : 2, "quizTotal" : 19, "labTotal" : 16, "examTotal" : 175 }
      - { "_id" : 3, "quizTotal" : 14, "labTotal" : 11, "examTotal" : 148 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Etapa *$project***
  - **Exemplo 1**
    - Tendo em conta o seguinte documento:

```
{
  "_id" : 1,
  title: "abc123",
  isbn: "0001122223334",
  author: { last: "zzz", first: "aaa" },
  copies: 5
}
```


- **Agregação - Etapa *$project***
  - **Exemplo 1**
    - Aplicando a seguinte etapa:
      - db.books.aggregate( [ { $project : { title : 1 , author : 1 } } ] )
    - É realizada a inclusão de campos específicos, neste caso, os campos ***_id,*** ***title*** e ***author.***
    - Obtendo o seguinte resultado:
      - { "_id" : 1, "title" : "abc123", "author" : { "last" : "zzz", "first" : "aaa" } }


- **Agregação - Etapa *$project***
  - **Exemplo 2**
    - Aplicando a seguinte etapa:
      - db.books.aggregate( [ { $project : { "author.first" : 0, "copies" : 0 } } ] )
      - equivalente a
      - db.books.aggregate( [ { $project: { "author": { "first": 0}, "copies" : 0 } } ] )
    - É realizada a exclusão de campos específicos em documentos embebidos, neste caso, os campos "***author.first***" e "***copies***", obtendo todos os outros campos não excluídos.

mongoDB®

- **Agregação - Etapa *$project***
  - **Exemplo 2**
    - Obtendo o seguinte resultado:

```
{
    "_id" : 1,
    "title" : "abc123",
    "isbn" : "0001122223334",
    "author" : {
        "last" : "zzz"
    },
    "copies" : 5,
}
```

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Etapa *$project***
  - **Exemplo 3**
    - Tendo em conta os seguintes documentos:

```
{
  "_id" : 2,
  title: "Baked Goods",
  isbn: "9999999999999",
  author: { last: "xyz", first: "abc", middle: "" },
  copies: 2,
  lastModified: "2017-07-21"
}
```

```
{
  "_id" : 3,
  title: "Ice Cream Cakes",
  isbn: "8888888888888",
  author: { last: "xyz", first: "abc", middle: "mmm" },
  copies: 5,
  lastModified: "2017-07-22"
}
```

mongoDB®

- **Agregação - Etapa *$project***
  - **Exemplo 3**
    - Aplicando a seguinte etapa:

```
db.books.aggregate( [
   {
      $project: {
         title: 1,
         "author.first": 1,
         "author.last" : 1,
         "author.middle": {
            $cond: {
               if: { $eq: [ "", "$author.middle" ] },
               then: "$$REMOVE",
               else: "$author.middle"
            }
         }
      }
   }
] )
```


- **Agregação - Etapa *$project***
  - **Exemplo 3**
    - É removido o campo “aninhado”, ***author.middle***, se o valor do campo for igual a “”, utilizando para o efeito a variável de sistema ***REMOVE***.
    - Obtendo o seguinte resultado:
      - { "_id" : 2, "title" : "Baked Goods", "author" : { "last" : "xyz", "first" : "abc" } }
      - { "_id" : 3, "title" : "Ice Cream Cakes", "author" : { "last" : "xyz", "first" : "abc", "middle" : "mmm" } }


- **Agregação - Etapa *$project***
  - **Exemplo 4**
    - Tendo em conta os seguintes documentos:
      - { _id: 1, user: "1234", stop: { title: "book1", author: "xyz", page: 32 } }
      - { _id: 2, user: "7890", stop: [ { title: "book2", author: "abc", page: 5 }, { title: "book3", author: "ijk", page: 100 } ] }


- **Agregação - Etapa *$project***
  - **Exemplo 4**
    - Aplicando a seguinte etapa:
      - db.bookmarks.aggregate( [ { $project: { "stop.title": 1 } } ] )
      - ou
      - db.bookmarks.aggregate( [ { $project: { stop: { title: 1 } } } ] )
    - É realizada a inclusão dos campos específicos em documentos embebidos.


- **Agregação - Etapa *$project***
  - **Exemplo 4**
    - Obtendo o seguinte resultado:
      - { "_id" : 1, "stop" : { "title" : "book1" } }
      - { "_id" : 2, "stop" : [ { "title" : "book2" }, { "title" : "book3" } ] }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Etapa *$project***
  - **Exemplo 5**
    - Tendo em conta o seguinte documento:

```
{
  "_id" : 1,
  title: "abc123",
  isbn: "0001122223334",
  author: { last: "zzz", first: "aaa" },
  copies: 5
}
```


- **Agregação - Etapa *$project***
  - **Exemplo 5**
    - Aplicando a seguinte etapa:
      - db.books.aggregate([{ $project: {

        title: 1,

        isbn: {

          prefix: { $substr: [ "$isbn", 0, 3 ] },

          group: { $substr: [ "$isbn", 3, 2 ] }

        }, lastName: "$author.last", copiesSold: "$copies" }}])

- **Agregação - Etapa *$project***
  - **Exemplo 5**
    - É realizada a inclusão de novos campos, sendo que o campo ***isbn*** é computado para um subdocumento com dois campos “aninhados”.
    - Obtendo o seguinte resultado:
      - { "_id" : 1, "title" : "abc123", "isbn" : { "prefix" : "000", "group" : "11" },"lastName" : "zzz","copiesSold" : 5 }


- **Agregação - Etapa *$project***
  - **Exemplo 6**
    - Tendo em conta o seguinte documento:
      - { "_id" : ObjectId("55ad167f320c6be244eb3b95"), "x" : 1, "y" : 1 }
    - Aplicando a etapa:
      - db.collection.aggregate( [ { $project: { myArray: [ "$x", "$y" ] } } ] )
    - É criado um novo campo com o nome "***myArray***" em que os elementos são os valores dos campos "***x***" e "***y***".


- **Agregação - Etapa *$project***
  - **Exemplo 6**
    - Obtendo o seguinte resultado:
      - { "_id" : ObjectId("55ad167f320c6be244eb3b95"), "myArray" : [ 1, 1 ] }


- **Agregação - Etapa *$project***
  - **Exemplo 7**
    - Se a especificação do *array* incluir campos não existentes, o valor do elemento colocado no *array* será ***null***.
    - Aplicando a seguinte etapa:
      - db.collection.aggregate( [ { $project: { myArray: [ "$x", "$y", "$someField" ] } } ] )
    - O resultado será:
      - { "_id" : ObjectId("55ad167f320c6be244eb3b95"), "myArray" : [ 1, 1, null ] }


- **Agregação - Etapa *$sort***
  - Ordena todos os documentos de entrada e retorna-os de forma ordenada.
  - Tem a seguinte sintaxe:
    - { $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }
  - A etapa trata do documento com os campos a serem ordenados.

- **Agregação - Etapa *$sort***
  - Por exemplo, o <sort order> pode ter os seguintes valores:
    - 1 para especificar ordem ascendente.
    - -1 para especificar ordem descendente.
  - **Nota**: existe mais um valor existente que podem consultar na documentação.


- **Agregação - Etapa *$sort***
  - **Exemplo 1**
    - db.users.aggregate( [ { $sort : { age : -1, posts: 1 } } ] )
    - A operação ordena todos os documentos na colecção, em ordem descendente de acordo com o campo ***age*** e depois em ordem ascendente consoante o campo ***posts***.

- **Agregação - Etapa *$limit***
  - Passa apenas o número de documentos especificado (limite máximo) para a próxima etapa do *pipeline*.
  - Não altera o conteúdo dos documentos.
  - Tem a seguinte sintaxe:
    - { $limit: <positive integer> }
    - <positive integer> aceita um valor inteiro positivo do número de documentos máximo a passar.


- **Agregação - Etapa *$limit***
  - **Exemplo 1**
    - db.pessoas.aggregate(

      { $limit : 5 }

      );
    - Retorna apenas os primeiros 5 documentos da colecção ***pessoas***.

- **Agregação - Etapa *$unwind***
  - Desconstrói um campo *array* de documentos de entrada, para criar um documento de saída, para cada elemento do *array.*
  - Cada documento de saída substitui o *array,* com um valor de elemento para cada documento de entrada.
  - Existem **n** documentos de saída, em que **n** é o número de elementos do *array*.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$unwind***
  - Tem duas sintaxes:
    - O operando é um caminho:
      - { $unwind: <field path> }
    - O operando é um documento:
      - { $unwind: {

        path: <field path>,

        includeArrayIndex: <string>,

        preserveNullAndEmptyArrays: <boolean> } }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$unwind***
  - Para especificar o <field path>:
    - Colocar o caractere **$** seguido do nome do campo único ou "aninhado" dentro de aspas duplas.

| **Campo** | **Tipo** | **Descrição** |
|---|---|---|
| path | *string* | O caminho para um campo *array*. Para especificar o caminho, necessário colocar o caractere **$** antes do nome do campo dentro de aspas duplas. |
| includeArrayIndex | *string* | Opcional. O nome do novo campo para guardar o índice do *array* do elemento. O nome não pode começar com o caractere **$**. |

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

- **Agregação - Etapa *$unwind***
  - Para especificar o <field path>:

| **Campo** | **Tipo** | **Descrição** |
|---|---|---|
| preserveNullAndEmptyArrays | booleano | Opcional. Se for ***true*** e o caminho (***path***) for ***null***, em falta ou *array* vazio, é realizada a saída do documento.<br>Se for *false*, não é realizada a saída do documento se o caminho (***path***) for ***null***, em falta ou ***array*** vazio.<br><br>Por omissão é ***false***. |

mongoDB®

- **Agregação - Etapa *$unwind***
  - **Notas**
    - Se o campo especificado, não é um *array* mas existe, não é ***null*** ou um *array* vazio, o valor desse campo é tratado como um *array* de apenas um elemento.
    - Se o campo especificado, não existe no documento ou o campo é um *array* vazio, por omissão é ignorado esse documento de entrada e não existirá documentos de saída para aquele documento de entrada.


- **Agregação - Etapa *$unwind***
  - **Exemplo 1**
    - Considerando o seguinte documento:
      - { "_id" : 1, "item" : "ABC1", sizes: [ "S", "M", "L"] }
    - Aplicando a etapa seguinte:
      - db.inventory.aggregate( [ { $unwind : "$sizes" } ] )


- **Agregação - Etapa *$unwind***
  - **Exemplo 1**
    - Serão obtidos três documentos correspondentes ao documento colocado anteriormente, em que cada um, tem apenas um elemento do *array*.
    - Obtendo o seguinte resultado:
      - { "_id" : 1, "item" : "ABC1", "sizes" : "S" }
      - { "_id" : 1, "item" : "ABC1", "sizes" : "M" }
      - { "_id" : 1, "item" : "ABC1", "sizes" : "L" }

mongoDB®

- **Agregação - Etapa *$unwind***
  - **Exemplo 2**
    - Tendo em conta a seguinte colecção:

```
{ "_id" : 1, "item" : "ABC", "sizes": [ "S", "M", "L"] }
{ "_id" : 2, "item" : "EFG", "sizes" : [ ] }
{ "_id" : 3, "item" : "IJK", "sizes": "M" }
{ "_id" : 4, "item" : "LMN" }
{ "_id" : 5, "item" : "XYZ", "sizes" : null }
```


- **Agregação - Etapa *$unwind***
  - **Exemplo 2**
    - Aplicando a seguinte etapa:
      - db.inventory.aggregate( [ { $unwind: "$sizes" } ] )
      - equivalente a
      - db.inventory.aggregate( [ { $unwind: { path: "$sizes" } } ] )
    - Se o caminho para o campo, não é um *array* mas existe, não é ***null***, ou *array* vazio, esse campo é tratado como um *array,* de apenas um elemento.


- **Agregação - Etapa *$unwind***
  - **Exemplo 2**
    - Obtendo o seguinte resultado:
      - { "_id" : 1, "item" : "ABC", "sizes" : "S" }
      - { "_id" : 1, "item" : "ABC", "sizes" : "M" }
      - { "_id" : 1, "item" : "ABC", "sizes" : "L" }
      - { "_id" : 3, "item" : "IJK", "sizes" : "M" }
    - Para incluir todos os documentos, em que o campo está em falta, é ***null*** ou *array* vazio é necessário utilizar a opção:
      - ***preserveNullAndEmptyArrays: true***

- **Agregação - Etapa *$out***
  - Escreve os documentos resultantes da agregação numa colecção.
  - Para ser utilizado tem de ser a última etapa do *pipeline*.

mongoDB®

- **Agregação - Etapa *$out***
  - Algumas notas:
    - Se existir esquema de validação, os documentos resultantes têm de obedecer às regras.
    - Para "ultrapassar" esta validação existe a possibilidade de activar a opção ***bypassDocumentValidation*** no comando ***aggregate***.
    - Se a colecção não existir, a mesma é criada.
    - A colecção só fica visível no fim do processo de agregação.
    - Se a agregação falhar a colecção não é criada.


- **Agregação - Etapa *$out***
  - Algumas notas:
    - Se a colecção já existir, a colecção existente é substituída pela nova colecção.
    - A agregação falha se os documentos produzidos pelo mesmo, violam a regra dos índices únicos, incluindo o índice do campo ***_id***.


- **Agregação - Etapa *$out***
  - Tem a seguinte sintaxe:
    - { $out: "<output-collection>" }
  - Não é possível especificar uma colecção ***sharded***. No entanto, a colecção de entrada no *pipeline* pode ser ***sharded***.
  - Não pode ser uma colecção ***capped.***
    - ***Capped ->*** são colecções de tamanho fixo, que funcionam como um ***buffer***, quando a colecção está cheia, os documentos antigos são sobrepostos pelos novos documentos na coleccção.

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Etapa *$out***
  - **Exemplo 1**
    - Tendo em conta os seguintes documentos:

```
{ "_id" : 8751, "title" : "The Banquet", "author" : "Dante", "copies" : 2 }
{ "_id" : 8752, "title" : "Divine Comedy", "author" : "Dante", "copies" : 1 }
{ "_id" : 8645, "title" : "Eclogues", "author" : "Dante", "copies" : 2 }
{ "_id" : 7000, "title" : "The Odyssey", "author" : "Homer", "copies" : 10 }
{ "_id" : 7020, "title" : "Iliad", "author" : "Homer", "copies" : 10 }
```


- **Agregação - Etapa *$out***
  - **Exemplo 1**
    - Aplicando a seguinte etapa:

```
db.books.aggregate( [
                    { $group : { _id : "$author", books: { $push: "$title" } } },
                    { $out : "authors" }
                  ] )
```


- **Agregação - Etapa *$out***
  - **Exemplo 1**
    - São agrupados os títulos (“***title***”) dos livros por autores (“***author***”).
    - Será criada uma colecção ***authors*** com os documentos resultantes:
      - { "_id" : "Homer", "books" : [ "The Odyssey", "Iliad" ] }
      - { "_id" : "Dante", "books" : [ "The Banquet", "Divine Comedy", "Eclogues" ] }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Exemplos globais**
  - **Exemplo 1**
    - Tendo em conta os seguintes documentos:

```
1, cust_id: "abc1", ord_date: ISODate("2012-11-02T17:04:11.102Z"), status: "A", amount: 50
2, cust_id: "xyz1", ord_date: ISODate("2013-10-01T17:04:11.102Z"), status: "A", amount: 100
3, cust_id: "xyz1", ord_date: ISODate("2013-10-12T17:04:11.102Z"), status: "D", amount: 25
4, cust_id: "xyz1", ord_date: ISODate("2013-10-11T17:04:11.102Z"), status: "D", amount: 125
5, cust_id: "abc1", ord_date: ISODate("2013-11-12T17:04:11.102Z"), status: "A", amount: 25
```


- **Agregação - Exemplos globais**
  - **Exemplo 1**
    - Aplicando a seguinte etapa:
      - db.orders.aggregate([

```
{ $match: { status: "A" } },
{ $group: { _id: "$cust_id", total: { $sum: "$amount" } } },
{ $sort: { total: -1 } }
])
```


- **Agregação - Exemplos globais**
  - **Exemplo 1**
    - Serão seleccionados apenas os documentos com o "***status***" igual a "**A**", agrupando-os pelo campo ***cust_id***, calculando o total para cada ***cust_id,*** a partir da soma do campo ***amount,*** no fim, ordena os resultados pelo campo total em ordem descendente.
    - Obtendo o resultado final:
      - { "_id" : "xyz1", "total" : 100 }
      - { "_id" : "abc1", "total" : 75 }

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II

mongoDB®

- **Agregação - Exemplos globais**
  - **Exemplo 2**
    - Tendo em conta o seguinte documento:
    - Uma colecção de código postais.

```
{
  "_id": "10280",
  "city": "NEW YORK",
  "state": "NY",
  "pop": 5574,
  "loc": [
    -74.016323,
    40.710537
  ]
}
```


- **Agregação - Exemplos globais**
  - **Exemplo 2**
    - Aplicando a seguinte etapa:
      - db.zipcodes.aggregate( [

        { $group: { _id: "$state", totalPop: { $sum: "$pop" } } },

        { $match: { totalPop: { $gte: 10*1000*1000 } } }

        ] )


- **Agregação - Exemplos globais**
  - **Exemplo 2**
    - A etapa ***$group***, agrupa todos os documentos da colecção pelo campo estado ("***state***"), calcula o ***totalPop*** para cada estado ("***state***"), e gera um documento de saída para cada estado ("***state***").
    - Os novos documentos de saída tem dois campos: o campo ***_id*** e o campo ***totalPop***.
    - O campo ***_id*** contém o valor do estado ("***state***"). O campo ***totalPop*** é um campo calculado e que contém o total de população por cada estado ("***state***").


- **Agregação - Exemplos globais**
  - **Exemplo 2**
    - Na etapa ***$match***, obtém-se todos os estados ("***states***"), com população maior ou igual a 10 milhões.
    - Resultado final:
      - {
        "_id" : "AK",
        "totalPop" : 550043
        }

- **Agregação - Comparação entre *SQL* e *MongoDB***

| SQL | MongoDB |
|---|---|
| WHERE | $match |
| GROUP BY | $group |
| HAVING | $match |
| SELECT | $project |
| ORDER BY | $sort |
| LIMIT | $limit |
| SUM() | $sum |
| COUNT() | $sum e $sortByCount |
| join | $lookup |

UNIDADE CURRICULAR : SISTEMAS GESTORES DE BASES DE DADOS II


- **Referências**
  - https://docs.mongodb.com/manual/core/aggregation-pipeline/
  - **https://docs.mongodb.com/manual/meta/aggregation-quick-reference/**


UNIVERSIDADE da MADEIRA
CTeSP
CURSOS TÉCNICOS
SUPERIORES PROFISSIONAIS