Banco de dados NoSQL - Agregação

Prof. Gustavo Leitão

Agregação

- Utilizado para realizar agrupamento de dados

```
db.collection.aggregate( [ <stage1>, <stage2>, ... ] )
```

- Cada *stage* representa uma etapa do processamento executado sequencialmente

https://docs.mongodb.com/manual/meta/aggregation-quick-reference/

Stages
Vários operações possíveis!

`$bucket`	Categorizes incoming documents into groups, called buckets, based on a specified expression and bucket boundaries.
`$bucketAuto`	Categorizes incoming documents into a specific number of groups, called buckets, based on a specified expression. Bucket boundaries are automatically determined in an attempt to evenly distribute the documents into the specified number of buckets.
`$collStats`	Returns statistics regarding a collection or view.
`$count`	Returns a count of the number of documents at this stage of the aggregation pipeline.
`$facet`	Processes multiple aggregation pipelines within a single stage on the same set of input documents. Enables the creation of multi-faceted aggregations capable of characterizing data across multiple dimensions, or facets, in a single stage.
`$geoNear`	Returns an ordered stream of documents based on the proximity to a geospatial point. Incorporates the functionality of `$match`, `$sort`, and `$limit` for geospatial data. The output documents include an additional distance field and can include a location identifier field.
`$graphLookup`	Performs a recursive search on a collection. To each output document, adds a new array field that contains the traversal results of the recursive search for that document.
`$group`	Groups input documents by a specified identifier expression and applies the accumulator expression(s), if specified, to each group. Consumes all input documents and outputs one document per each distinct group. The output documents only contain the identifier field and, if specified, accumulated fields.
`$indexStats`	Returns statistics regarding the use of each index for the collection.
`$limit`	Passes the first *n* documents unmodified to the pipeline where *n* is the specified limit. For each input document, outputs either one document (for the first *n* documents) or zero documents (after the first *n* documents).

https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/

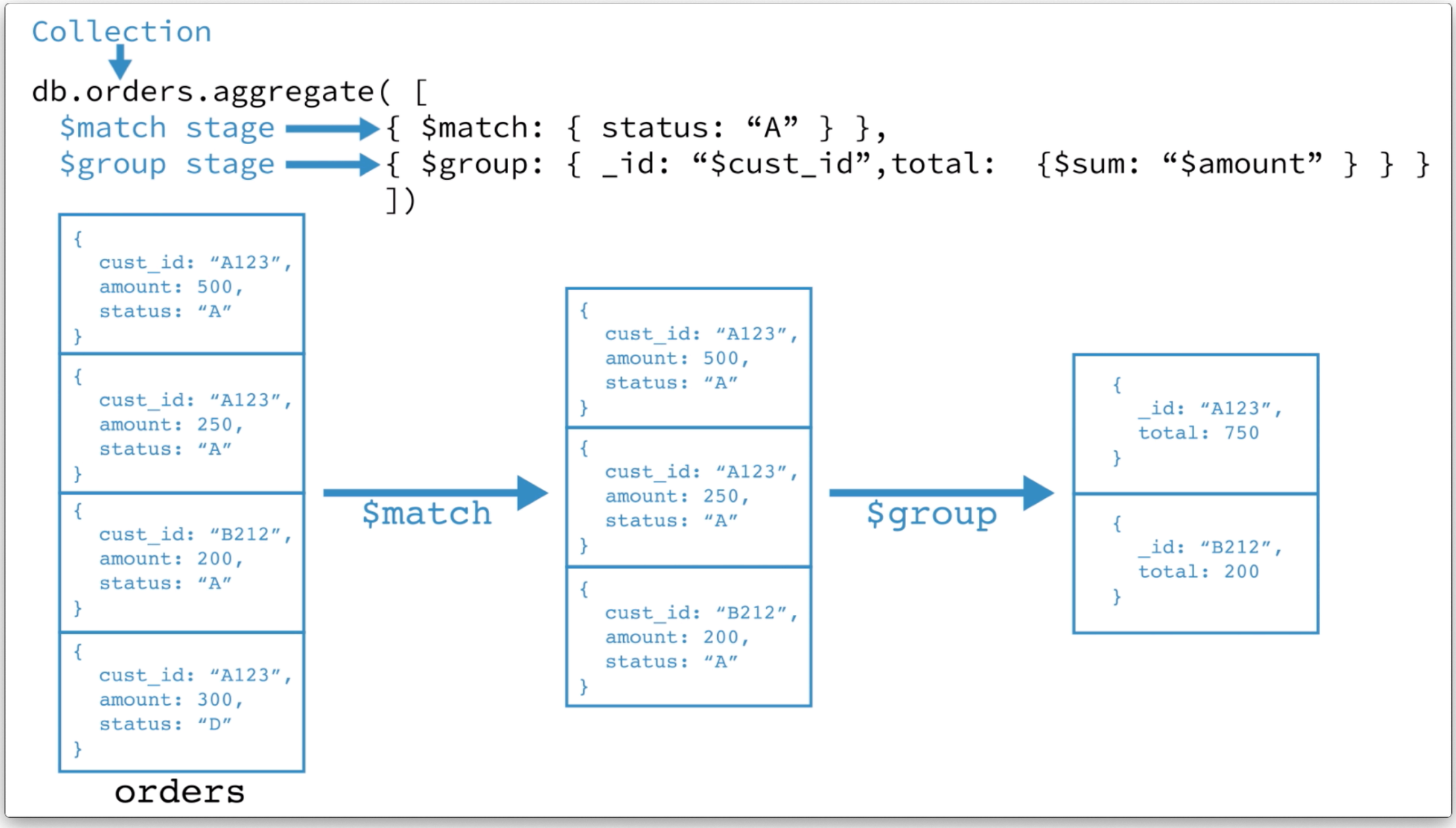
Stages
Vários operações possíveis!

`$bucket`	Categorizes incoming documents into groups, called buckets, based on a specified expression and bucket boundaries.
`$bucketAuto`	Categorizes incoming documents into a specific number of groups, called buckets, based on a specified expression. Bucket boundaries are automatically determined in an attempt to evenly distribute the documents into the specified number of buckets.
`$collStats`	Returns statistics regarding a collection or view.
`$count`	Returns a count of the number of documents at this stage of the aggregation pipeline.
`$facet`	Processes multiple aggregation pipelines within a single stage on the same set of input documents. Enables the creation of multi-faceted aggregations capable of characterizing data across multiple dimensions, or facets, in a single stage.
`$geoNear`	Returns an ordered stream of documents based on the proximity to a geospatial point. Incorporates the functionality of `$match`, `$sort`, and `$limit` for geospatial data. The output documents include an additional distance field and can include a location identifier field.
`$graphLookup`	Performs a recursive search on a collection. To each output document, adds a new array field that contains the traversal results of the recursive search for that document.
`$group`	Groups input documents by a specified identifier expression and applies the accumulator expression(s), if specified, to each group. Consumes all input documents and outputs one document per each distinct group. The output documents only contain the identifier field and, if specified, accumulated fields.
`$indexStats`	Returns statistics regarding the use of each index for the collection.
`$limit`	Passes the first *n* documents unmodified to the pipeline where *n* is the specified limit. For each input document, outputs either one document (for the first *n* documents) or zero documents (after the first *n* documents).

https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/

Agregação

Agregação é a operação que permite agrupar dados de múltiplos documentos.



Agrupa restaurantes por bairros totalizando o total por bairro

```
db.restaurants.aggregate(
   [
     { $group: { "_id": "$borough", "count": { $sum: 1 } } }
   ]
);
```

The result set consists of the following documents:

```
{ "_id" : "Staten Island", "count" : 969 }
{ "_id" : "Brooklyn", "count" : 6086 }
{ "_id" : "Manhattan", "count" : 10259 }
{ "_id" : "Queens", "count" : 5656 }
{ "_id" : "Bronx", "count" : 2338 }
{ "_id" : "Missing", "count" : 51 }
```

Exercício

- Localize a quantidade de restaurantes por *zipcode* de cozinha brasileira no bairro do Queens

Agrupa restaurantes que satisfazem condição

```
db.restaurants.aggregate(
    [
      { $match: { "borough": "Queens", "cuisine": "Brazilian" } },
      { $group: { "_id": "$address.zipcode" , "count": { $sum: 1 } } }
    ]
);
```

The result set consists of the following documents:

```
{ "_id" : "11368", "count" : 1 }
{ "_id" : "11106", "count" : 3 }
{ "_id" : "11377", "count" : 1 }
{ "_id" : "11103", "count" : 1 }
{ "_id" : "11101", "count" : 2 }
```

Operadores para agrupamento

Name	Description
`$addToSet`	Returns an array of *unique* expression values for each group. Order of the array elements is undefined.
`$avg`	Returns an average of numerical values. Ignores non-numeric values.
`$first`	Returns a value from the first document for each group. Order is only defined if the documents are in a defined order.
`$last`	Returns a value from the last document for each group. Order is only defined if the documents are in a defined order.
`$max`	Returns the highest expression value for each group.
`$mergeObjects`	Returns a document created by combining the input documents for each group.
`$min`	Returns the lowest expression value for each group.
`$push`	Returns an array of expression values for each group.
`$stdDevPop`	Returns the population standard deviation of the input values.
`$stdDevSamp`	Returns the sample standard deviation of the input values.
`$sum`	Returns a sum of numerical values. Ignores non-numeric values.

$avg

```
{ "_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") }
```

```
db.sales.aggregate(
    [
      {
        $group:
          {
            _id: "$item",
            avgAmount: { $avg: { $multiply: [ "$price", "$quantity" ] } },
            avgQuantity: { $avg: "$quantity" }
          }
      }
    ]
)
```

$avg

Retorna o valor médio da expressão para cada grupo.

```
db.sales.aggregate(
   [
     {
       $group:
         {
           _id: "$item",
           avgAmount: { $avg: { $multiply: [ "$price", "$quantity" ] } },
           avgQuantity: { $avg: "$quantity" }
         }
     }
   ]
)
```

Resultado

```
{ "_id" : "xyz", "avgAmount" : 37.5, "avgQuantity" : 7.5 }
{ "_id" : "jkl", "avgAmount" : 20, "avgQuantity" : 1 }
{ "_id" : "abc", "avgAmount" : 60, "avgQuantity" : 6 }
```

$max

Retorna o maior valor de expressão para cada grupo.

```
{ "_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") }
```

```
db.sales.aggregate(
    [
      {
        $group:
          {
            _id: "$item",
            maxTotalAmount: { $max: { $multiply: [ "$price", "$quantity" ] } },
            maxQuantity: { $max: "$quantity" }
          }
      }
    ]
)
```

$max

Retorna o maior valor de expressão para cada grupo.

```
db.sales.aggregate(
    [
      {
        $group:
          {
            _id: "$item",
            maxTotalAmount: { $max: { $multiply: [ "$price", "$quantity" ] } },
            maxQuantity: { $max: "$quantity" }
          }
      }
    ]
)
```

Resultado

```
{ "_id" : "xyz", "maxTotalAmount" : 50, "maxQuantity" : 10 }
{ "_id" : "jkl", "maxTotalAmount" : 20, "maxQuantity" : 1 }
{ "_id" : "abc", "maxTotalAmount" : 100, "maxQuantity" : 10 }
```

$min

Retorna o menor valor de expressão para cada grupo.

```
{ "_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") }
```

```
db.sales.aggregate(
    [
      {
        $group:
          {
            _id: "$item",
            minQuantity: { $min: "$quantity" }
          }
      }
    ]
)
```

$min

Retorna o menor valor de expressão para cada grupo.

```
db.sales.aggregate(
   [
     {
       $group:
         {
           _id: "$item",
           minQuantity: { $min: "$quantity" }
         }
     }
   ]
)
```

Resultado

```
{ "_id" : "xyz", "minQuantity" : 5 }
{ "_id" : "jkl", "minQuantity" : 1 }
{ "_id" : "abc", "minQuantity" : 2 }
```

$first

Retorna o primeiro valor resultante da aplicação de uma expressão ao último documento em um grupo de documentos que compartilham o mesmo grupo por um campo. Somente significativo quando os documentos estão em uma ordem definida.

```
db.sales.aggregate(
    [
      { $sort: { item: 1, date: 1 } },
      {
        $group:
          {
            _id: "$item",
            firstSalesDate: { $first: "$date" }
          }
      }
    ]
)
```

Resultado

```
{ "_id" : "xyz", "firstSalesDate" : ISODate("2014-02-03T09:05:00Z") }
{ "_id" : "jkl", "firstSalesDate" : ISODate("2014-02-03T09:00:00Z") }
{ "_id" : "abc", "firstSalesDate" : ISODate("2014-01-01T08:00:00Z") }
```

$last

Retorna o último valor resultante da aplicação de uma expressão ao último documento em um grupo de documentos que compartilham o mesmo grupo por um campo. Somente significativo quando os documentos estão em uma ordem definida.

```
db.sales.aggregate(
    [
      { $sort: { item: 1, date: 1 } },
      {
        $group:
          {
            _id: "$item",
            lastSalesDate: { $last: "$date" }
          }
      }
    ]
)
```

Resultado

```
{ "_id" : "xyz", "lastSalesDate" : ISODate("2014-02-15T14:12:12Z") }
{ "_id" : "jkl", "lastSalesDate" : ISODate("2014-02-03T09:00:00Z") }
{ "_id" : "abc", "lastSalesDate" : ISODate("2014-02-15T08:00:00Z") }
```

$addToSet

Retorna uma vetor com valores únicos resultantes da aplicação de uma expressão a cada documento em um grupo de documentos que compartilham o mesmo grupo por chave.

```
db.sales.aggregate(
   [
     {
       $group:
         {
           _id: { day: { $dayOfYear: "$date"}, year: { $year: "$date" } },
           itemsSold: { $addToSet: "$item" }
         }
     }
   ]
)
```

Resultado

```
{ "_id" : { "day" : 46, "year" : 2014 }, "itemsSold" : [ "xyz", "abc" ] }
{ "_id" : { "day" : 34, "year" : 2014 }, "itemsSold" : [ "xyz", "jkl" ] }
{ "_id" : { "day" : 1, "year" : 2014 }, "itemsSold" : [ "abc" ] }
```

$push

Retorna uma vetor com todos os valores resultantes da aplicação de uma expressão a cada documento em um grupo de documentos que compartilham o mesmo grupo por chave.

```
db.sales.aggregate(
   [
     {
       $group:
         {
           _id: { day: { $dayOfYear: "$date"}, year: { $year: "$date" } },
           itemsSold: { $push:  { item: "$item", quantity: "$quantity" } }
         }
     }
   ]
)
```

Resultado

```
{ "_id" : { "day" : 46, "year" : 2014 }, "itemsSold" : [ "xyz", "abc" ] }
{ "_id" : { "day" : 34, "year" : 2014 }, "itemsSold" : [ "xyz", "jkl" ] }
{ "_id" : { "day" : 1, "year" : 2014 }, "itemsSold" : [ "abc" ] }
```

$project

Stage que permite escolher campos de um documento a serem retornados

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

The following `$project` stage excludes the `_id` field but includes the `title`, and the `author` fields in its output documents:

```
db.books.aggregate( [ { $project : { _id: 0, title : 1 , author : 1 } } ] )
```

The operation results in the following document:

```
{ "title" : "abc123", "author" : { "last" : "zzz", "first" : "aaa" } }
```

Exercício

- Faça uma consulta para retornar a quantidade de restaurantes por letra inicial de seu nome. Traga os dados ordenados alfabeticamente A —> Z.

- Ex: {A: 1923, B: 4747, C: 4744}

Exercício

- Faça uma consulta para retornar a quantidade de restaurantes por letra inicial de seu nome. Traga os dados ordenados alfabeticamente A —> Z.

```
db.restaurants.aggregate([
  {$set: {"letra": {$substr: [ "$name", 0, 1 ]}}},
  {$group: {"_id": "$letra", "count": {$sum: 1}}},
  {$sort: { _id: 1 }}
])
```

$set

Adiciona campos ao documento

```
db.scores.insertMany([
   { _id: 1, student: "Maya", homework: [ 10, 5, 10 ], quiz: [ 10, 8 ], extraCredit: 0 },
   { _id: 2, student: "Ryan", homework: [ 5, 6, 5 ], quiz: [ 8, 8 ], extraCredit: 8 }
])
```

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

$set

Adiciona campos ao documento

```
{
  "_id" : 1,
  "student" : "Maya",
  "homework" : [ 10, 5, 10 ],
  "quiz" : [ 10, 8 ],
  "extraCredit" : 0,
  "totalHomework" : 25,
  "totalQuiz" : 18,
  "totalScore" : 43
}
{
  "_id" : 2,
  "student" : "Ryan",
  "homework" : [ 5, 6, 5 ],
  "quiz" : [ 8, 8 ],
  "extraCredit" : 8,
  "totalHomework" : 16,
  "totalQuiz" : 16,
  "totalScore" : 40
}
```

$sample

Obtém N documentos aleatórios

```
{ "_id" : 1, "name" : "dave123", "q1" : true, "q2" : true }
{ "_id" : 2, "name" : "dave2", "q1" : false, "q2" : false  }
{ "_id" : 3, "name" : "ahn", "q1" : true, "q2" : true  }
{ "_id" : 4, "name" : "li", "q1" : true, "q2" : false  }
{ "_id" : 5, "name" : "annT", "q1" : false, "q2" : true  }
{ "_id" : 6, "name" : "li", "q1" : true, "q2" : true  }
{ "_id" : 7, "name" : "ty", "q1" : false, "q2" : true  }
```

Sintaxe

```
db.users.aggregate(
   [ { $sample: { size: 3 } } ]
)
```

Pontos de atenção (para melhor performance):

- $sample deve ser o primeiro estágio do pipeline
- N é menor que 5% do total de documentos
- A coleção possui mais de 100 documentos

$unwind

Desconstrói um array criando um documento para cada item do array

```
db.inventory.insertOne({ "_id" : 1, "item" : "ABC1", sizes: [ "S", "M", "L"] })
```

Sintaxe

```
db.inventory.aggregate( [ { $unwind : "$sizes" } ] )
```

Resultado

```
{ "_id" : 1, "item" : "ABC1", "sizes" : "S" }
{ "_id" : 1, "item" : "ABC1", "sizes" : "M" }
{ "_id" : 1, "item" : "ABC1", "sizes" : "L" }
```

$unwind

Desconstrói um array criando um documento para cada item do array

```
db.inventory2.insertMany([
  { "_id" : 1, "item" : "ABC", price: NumberDecimal("80"), "sizes": [ "S", "M", "L"] },
  { "_id" : 2, "item" : "EFG", price: NumberDecimal("120"), "sizes" : [ ] },
  { "_id" : 3, "item" : "IJK", price: NumberDecimal("160"), "sizes": "M" },
  { "_id" : 4, "item" : "LMN" , price: NumberDecimal("10") },
  { "_id" : 5, "item" : "XYZ", price: NumberDecimal("5.75"), "sizes" : null }
])
```

Sintaxe

```
db.inventory2.aggregate( [ { $unwind: "$sizes" } ] )
db.inventory2.aggregate( [ { $unwind: { path: "$sizes" } } ] )
```

```
{ "_id" : 1, "item" : "ABC", "price" : NumberDecimal("80"), "sizes" : "S" }
{ "_id" : 1, "item" : "ABC", "price" : NumberDecimal("80"), "sizes" : "M" }
{ "_id" : 1, "item" : "ABC", "price" : NumberDecimal("80"), "sizes" : "L" }
{ "_id" : 3, "item" : "IJK", "price" : NumberDecimal("160"), "sizes" : "M" }
```

$unwind

Desconstrói um array criando um documento para cada item do array

```
db.inventory2.aggregate( [
   { $unwind: { path: "$sizes", preserveNullAndEmptyArrays: true } }
] )
```

Resultado

```
{ "_id" : 1, "item" : "ABC", "price" : NumberDecimal("80"), "sizes" : "S" }
{ "_id" : 1, "item" : "ABC", "price" : NumberDecimal("80"), "sizes" : "M" }
{ "_id" : 1, "item" : "ABC", "price" : NumberDecimal("80"), "sizes" : "L" }
{ "_id" : 2, "item" : "EFG", "price" : NumberDecimal("120") }
{ "_id" : 3, "item" : "IJK", "price" : NumberDecimal("160"), "sizes" : "M" }
{ "_id" : 4, "item" : "LMN", "price" : NumberDecimal("10") }
{ "_id" : 5, "item" : "XYZ", "price" : NumberDecimal("5.75"), "sizes" : null }
```

Exercício

- Faça uma consulta que totalize o total de avaliações com grade A
- Faça uma consulta para totalizar o número de avaliações com grade por grade.

Exercício

- Faça uma consulta que totalize o total de avaliações com grade A

```
db.restaurants.aggregate([
     {$unwind: "$grades"},
     {$match: {"grades.grade": 'A'}},
     {$group: {_id: null, total: {$sum: 1}}}
])
```

- Faça uma consulta para totalizar o número de avaliações com grade por grade.

```
db.restaurants.aggregate([
     {$unwind: "$grades"},
     {$group: {_id: "$grades.grade", total: {$sum: 1}}}
])
```

$mergeObjects

Mescla dois documentos em um só

Example	Results
`{ $mergeObjects: [ { a: 1 }, null ] }`	`{ a: 1 }`
`{ $mergeObjects: [ null, null ] }`	`{ }`
`{ $mergeObjects: [ { a: 1 }, { a: 2, b: 2 }, { a: 3, c: 3 } ] }`	`{ a: 3, b: 2, c: 3 }`
`{ $mergeObjects: [ { a: 1 }, { a: 2, b: 2 }, { a: 3, b: null, c: 3 } ] }`	`{ a: 3, b: null, c: 3 }`

$mergeObjects

Mescla dois ou mais documentos em um só

```
db.sales.insert( [
   { _id: 1, year: 2017, item: "A", quantity: { "2017Q1": 500, "2017Q2": 500 } },
   { _id: 2, year: 2016, item: "A", quantity: { "2016Q1": 400, "2016Q2": 300,
"2016Q3": 0, "2016Q4": 0 } } ,
   { _id: 3, year: 2017, item: "B", quantity: { "2017Q1": 300 } },
   { _id: 4, year: 2016, item: "B", quantity: { "2016Q3": 100, "2016Q4": 250 } }
] )
```

```
db.sales.aggregate( [
   { $group: { _id: "$item", mergedSales: { $mergeObjects: "$quantity" } } }
])
```

Resultado:

```
{ "_id" : "B", "mergedSales" : { "2017Q1" : 300, "2016Q3" : 100, "2016Q4" :
250 } }
{ "_id" : "A", "mergedSales" : { "2017Q1" : 500, "2017Q2" : 500, "2016Q1" : 400,
"2016Q2" : 300, "2016Q3" : 0, "2016Q4" : 0 } }
```

$replaceRoot

Troca um documento por outro. Todos os campos são trocados, inclusive o _id.

```
db.people.insertMany([
     { "_id" : 1, "name" : "Arlene", "age" : 34, "pets" : { "dogs" : 2, "cats" : 1 } }
     { "_id" : 2, "name" : "Sam", "age" : 41, "pets" : { "cats" : 1, "fish" : 3 } }
     { "_id" : 3, "name" : "Maria", "age" : 25 }
])
```

Exemplo:

```
db.people.aggregate( [
   { $replaceRoot: { newRoot: { $mergeObjects:  [ { dogs: 0, cats: 0, birds: 0, fish: 0
}, "$pets" ] }} }
] )
```

O mergeObjects irá mesclar o documento *default* com $pets

Resultado:

```
{ "dogs" : 2, "cats" : 1, "birds" : 0, "fish" : 0 }
{ "dogs" : 0, "cats" : 1, "birds" : 0, "fish" : 3 }
{ "dogs" : 0, "cats" : 0, "birds" : 0, "fish" : 0 }
```

$lookup

Realiza junção de elementos de outras coleções no documento retornado.

Sintaxe:

copy

```
{
   $lookup:
     {
       from: <collection to join>,
       localField: <field from the input documents>,
       foreignField: <field from the documents of the "from" collection>,
       as: <output array field>
     }
}
```

$lookup

Realiza junção de elementos de outras coleções no documento retornado.

Considere:

```
db.orders.insert([
   { "_id" : 1, "item" : "almonds", "price" : 12, "quantity" : 2 },
   { "_id" : 2, "item" : "pecans", "price" : 20, "quantity" : 1 },
   { "_id" : 3  }
])
```

```
db.inventory.insert([
   { "_id" : 1, "sku" : "almonds", description: "product 1", "instock" : 120 },
   { "_id" : 2, "sku" : "bread", description: "product 2", "instock" : 80 },
   { "_id" : 3, "sku" : "cashews", description: "product 3", "instock" : 60 },
   { "_id" : 4, "sku" : "pecans", description: "product 4", "instock" : 70 },
   { "_id" : 5, "sku": null, description: "Incomplete" },
   { "_id" : 6 }
])
```

$lookup

Realiza junção de elementos de outras coleções no documento retornado.

Comando:

```
db.orders.aggregate([
    {
      $lookup:
        {
          from: "inventory",
          localField: "item",
          foreignField: "sku",
          as: "inventory_docs"
        }
    }
])
```

copy

```
{
   "_id" : 1,
   "item" : "almonds",
   "price" : 12,
   "quantity" : 2,
   "inventory_docs" : [
      { "_id" : 1, "sku" : "almonds", "description" : "product 1", "instock" : 120 }
   ]
}
{
   "_id" : 2,
   "item" : "pecans",
   "price" : 20,
   "quantity" : 1,
   "inventory_docs" : [
      { "_id" : 4, "sku" : "pecans", "description" : "product 4", "instock" : 70 }
   ]
}
{
   "_id" : 3,
   "inventory_docs" : [
      { "_id" : 5, "sku" : null, "description" : "Incomplete" },
      { "_id" : 6 }
   ]
}
```

$lookup

Realiza junção de elementos de outras coleções no documento retornado.

Uso com arrays.

```
db.classes.insert( [
   { _id: 1, title: "Reading is ...", enrollmentlist: [ "giraffe2", "pandabear",
"artie" ], days: ["M", "W", "F"] },
   { _id: 2, title: "But Writing ...", enrollmentlist: [ "giraffe1", "artie" ],
days: ["T", "F"] }
])
```

```
db.members.insert( [
   { _id: 1, name: "artie", joined: new Date("2016-05-01"), status: "A" },
   { _id: 2, name: "giraffe", joined: new Date("2017-05-01"), status: "D" },
   { _id: 3, name: "giraffe1", joined: new Date("2017-10-01"), status: "A" },
   { _id: 4, name: "panda", joined: new Date("2018-10-11"), status: "A" },
   { _id: 5, name: "pandabear", joined: new Date("2018-12-01"), status: "A" },
   { _id: 6, name: "giraffe2", joined: new Date("2018-12-01"), status: "D" }
])
```

$lookup

Realiza junção de elementos de outras coleções no documento retornado.

Uso com arrays.

```
db.classes.aggregate([
   {
      $lookup:
         {
            from: "members",
            localField: "enrollmentlist",
            foreignField: "name",
            as: "enrollee_info"
         }
   }
])
```

$lookup

Realiza junção de elementos de outras coleções no documento retornado.

Uso com arrays.

```
{
   "_id" : 1,
   "title" : "Reading is ...",
   "enrollmentlist" : [ "giraffe2", "pandabear", "artie" ],
   "days" : [ "M", "W", "F" ],
   "enrollee_info" : [
      { "_id" : 1, "name" : "artie", "joined" : ISODate("2016-05-01T00:00:00Z"), "status" : "A" },
      { "_id" : 5, "name" : "pandabear", "joined" : ISODate("2018-12-01T00:00:00Z"), "status" : "A" },
      { "_id" : 6, "name" : "giraffe2", "joined" : ISODate("2018-12-01T00:00:00Z"), "status" : "D" }
   ]
}
{
   "_id" : 2,
   "title" : "But Writing ...",
   "enrollmentlist" : [ "giraffe1", "artie" ],
   "days" : [ "T", "F" ],
   "enrollee_info" : [
      { "_id" : 1, "name" : "artie", "joined" : ISODate("2016-05-01T00:00:00Z"), "status" : "A" },
      { "_id" : 3, "name" : "giraffe1", "joined" : ISODate("2017-10-01T00:00:00Z"), "status" : "A" }
   ]
}
```

$lookup e $mergeObjects

Considere:

```
db.orders.insert([
   { "_id" : 1, "item" : "almonds", "price" : 12, "quantity" : 2 },
   { "_id" : 2, "item" : "pecans", "price" : 20, "quantity" : 1 }
])
```

```
db.items.insert([
  { "_id" : 1, "item" : "almonds", description: "almond clusters", "instock" : 120 },
  { "_id" : 2, "item" : "bread", description: "raisin and nut bread", "instock" : 80 },
  { "_id" : 3, "item" : "pecans", description: "candied pecans", "instock" : 60 }
])
```

$lookup e $mergeObjects

```
db.orders.aggregate([
   {
      $lookup: {
         from: "items",
         localField: "item",    // field in the orders collection
         foreignField: "item",  // field in the items collection
         as: "fromItems"
      }
   }
])
```

Resultado:

```
{
  "_id" : 1,
  "item" : "almonds",
  "price" : 12,
  "quantity" : 2,
  "fromItems" : [
    {
      "_id" : 1,
      "item" : "almonds",
      "description" : "almond clusters",
      "instock" : 120
    }
  ]
}
…
```

$lookup e $mergeObjects

```
db.orders.aggregate([
   {
      $lookup: {
         from: "items",
         localField: "item",    // field in the orders collection
         foreignField: "item",  // field in the items collection
         as: "fromItems"
      }
   },{
      $replaceRoot: { newRoot: { $mergeObjects: [ { $arrayElemAt: [ "$fromItems", 0 ] }, "$$ROOT" ] } }
   }])
```

Resultado:

```
{
  "_id" : 1,
  "item" : "almonds",
  "description" : "almond clusters",
  "instock" : 120,
  "price" : 12,
  "quantity" : 2,
  "fromItems" : [
    {
      "_id" : 1,
      "item" : "almonds",
      "description" : "almond clusters",
      "instock" : 120
    }
  ]
}

…
```

$lookup e $mergeObjects

```
db.orders.aggregate([
   {
      $lookup: {
         from: "items",
         localField: "item",    // field in the orders collection
         foreignField: "item",  // field in the items collection
         as: "fromItems"
      }
   },
   {
      $replaceRoot: { newRoot: { $mergeObjects: [ { $arrayElemAt: [ "$fromItems", 0 ] }, "$$ROOT" ] } }
   },
   { $project: { fromItems: 0 } }
])
```

```
{ "_id" : 1, "item" : "almonds", "description" : "almond clusters", "instock" : 120, "price" : 12, "quantity" : 2 }
{ "_id" : 2, "item" : "pecans", "description" : "candied pecans", "instock" : 60, "price" : 20, "quantity" : 1 }
```

Exercício

- Faça uma consulta que totalize o total de avaliações com grade A

```
db.restaurants.aggregate([
     {$unwind: "$grades"},
     {$match: {"grades.grade": 'A'}},
     {$group: {_id: null, total: {$sum: 1}}}
])
```

- Faça uma consulta para totalizar o número de avaliações com grade por grade.

```
db.restaurants.aggregate([
     {$unwind: "$grades"},
     {$group: {_id: "$grades.grade", total: {$sum: 1}}}
])
```

Banco de dados NoSQL - MongoDB (Parte2)

Prof. Gustavo Leitão