
UNITAU
Universidade de Taubaté

MongoDB Agregação

Agregação

```
> db.catalogo.insertMany([
    { "_id": 1, "item": "TV", "tipo":
"Eletronicos", "preco": 2500.00, "taxas":
520.50, "desconto": 10.0, "dimensoes":
[ { "k": "L", "v": 1116.8}, { "k": "A", "v":
664.2}, { "k": "P", "v": 59.9}, { "k": "um",
"v": "mm" } ] },
    { "_id": 2, "item": "DVD Player", "tipo":
"Eletronicos", "preco": 250.00, "taxas":
30.00, "desconto": 10.0, "dimensoes":
[ [ "A", 16.0 ], [ "L", 63.0 ], [ "P",
46.0 ], [ "um", "cm" ] ] },
    { "_id": 3, "item": "SoundBar", "tipo":
"Eletronicos", "preco": 1080.00, "taxas":
120.00, "dimensoes": [ [ "L", 85.5 ], [ "A",
14.5 ], [ "P", 12.5 ], [ "um", "cm" ] ] },
```

Agregação

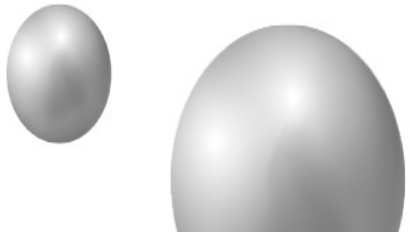

```
    { "_id": 4, "item": "Guarda Roupa",
"tipo": "Moveis", "preco": 2300.00, "taxas":
300.00, "desconto": 15.0, "dimensoes":
[ [ "L", 2 ], [ "A", 235.0 ], [ "L", 274.0 ],
[ "P", 55.5 ], [ "um", "cm" ] ] },
    { "_id": 5, "item": "Sofa", "tipo":
"Moveis", "preco": 1800.00, "taxas": 210.00,
"desconto": 10.0, "dimensoes": [ [ "A",
0.98 ], [ "L", 2.10 ], [ "P", 1.18 ], [ "um",
"m" ] ] },
    { "_id": 6, "item": "Mesa", "tipo":
"Moveis", "preco": 1200.00, "taxas": 140.00,
"desconto": 5.0, "dimensoes": [ [ "L",
80.0 ], [ "A", 76.5 ], [ "P", 140.0 ],
[ "um", "cm" ] ] },
```

Agregação

```
    { "_id": 7, "item": "Impressora", "tipo":
"Informatica", "preco": 520.00, "taxas":
60.00, "desconto": 7.5, "dimensoes": [ [ "L",
42.5 ], [ "A", 15.4 ], [ "P", 30.4 ], [ "um",
"cm" ] ] },
    { "_id": 8, "item": "Notebook", "tipo":
"Informatica", "preco": 2900.00, "taxas":
320.00, "dimensoes": [ [ "A", 1.99 ], [ "L",
38.0 ], [ "P", 25.8 ], [ "um", "cm" ] ] },
    { "_id": 9, "item": "Monitor", "tipo":
"Informatica", "preco": 570.00, "taxas":
65.00, "desconto": 8.0, "dimensoes": [ [ "L",
507.0 ], [ "A", 390.0 ], [ "P", 20.0 ],
[ "um", "mm" ] ] }
    ])
```

Agregação

```
{
        "acknowledged" : true,
        "insertedIds" : [
                1,
                2,
                3,
                4,
                5,
                6,
                7,
                8,
                9
        ]
}
```



Estágios de Agregação: $project

$project determina os campos que serão retornados no documento.

```
{ $project: { <specification(s)> } }
```

A especificação da projeção de cada campo pode ser:

- `<field>: <1 or true>` - inclusão do campo
- `_id: <0 or false>` - exclusão do campo _id
- `<field>: <expression>` - adiciona ou altera o valor do campo, também permite renomear um campo
- `<field>:<0 or false>` - exclusão do campo

Se for especificada a exclusão de um campo que não seja o campo _id, não será permitido o uso de outros tipos de especificação (inclusão de campo ou atribuição de valor) na projeção.

Estágios de Agregação: $project

```
> db.produtos.aggregate(
    {$project: {_id: 0,
                item: 1,
                qtd: "$quantidade",
                min: "$minimo"}}
  )
{ "item" : "envelope", "qtd" : 20, "min" : 10
}
{ "item" : "selos", "qtd" : 30, "min" :
null }
{ "item" : "lapis", "qtd" : 50, "min" : 30 }
{ "item" : "cartao", "qtd" : 15, "min" : 5 }
{ "item" : "caderno", "qtd" : 20 }
{ "item" : "borracha", "qtd" : 25, "min" : 10
}
```

Estágios de Agregação: $project

Campos em documentos embutidos podem ser referenciados pela notação de ponto ou notação de campos embutidos.

```
> db.editora.aggregate( {$project: {nome: 1,
"sede.pais": 1}} )
{ "_id" : "oreilly", "nome" : "O'Reilly
Media", "sede" : { "pais" : "Estados
Unidos" } }
{ "_id" : "cm", "nome" : "Editora Ciencia
Moderna", "sede" : { "pais" : "Brasil" } }
{ "_id" : "novatec", "nome" : "Novatec
Editora Ltda", "sede" : { "pais" : "Brasil" }
}
```

Estágios de Agregação: $project

```
> db.editora.aggregate( {$project: {nome: 1, sede: {pais: 1}}} )
{ "_id" : "oreilly", "nome" : "O'Reilly Media", "sede" : { "pais" : "Estados Unidos" } }
{ "_id" : "cm", "nome" : "Editora Ciencia Moderna", "sede" : { "pais" : "Brasil" } }
{ "_id" : "novatec", "nome" : "Novatec Editora Ltda", "sede" : { "pais" : "Brasil" } }
}
```

Estágios de Agregação: $project

Campos de documentos embutidos podem ser movidos renomeando o campo.

```
> db.editora.aggregate( {$project: {nome: 1,
pais: "$sede.pais"}} )
{ "_id" : "oreilly", "nome" : "O'Reilly
Media", "pais" : "Estados Unidos" }
{ "_id" : "novatec", "nome" : "Novatec
Editora Ltda", "pais" : "Brasil" }
{ "_id" : "cm", "nome" : "Editora Ciencia
Moderna", "pais" : "Brasil" }
```

Estágios de Agregação: $project

É possível adicionar campos a documentos embutidos mas o documento será sobrescrito com os campos relacionados. Para apenas adicionar um novo campo é mais fácil utilizar o estágio $addFields.

```
> db.editora.aggregate( {$project: {nome: 1,
"sede.telefone": "n/c"}} )
{ "_id" : "oreilly", "nome" : "O'Reilly
Media", "sede" : { "telefone" : "n/c" } }
{ "_id" : "novatec", "nome" : "Novatec
Editora Ltda", "sede" : { "telefone" :
"n/c" } }
{ "_id" : "cm", "nome" : "Editora Ciencia
Moderna", "sede" : { "telefone" : "n/c" } }
```

Estágios de Agregação: $project

Como o estágio de projeção é aplicado individualmente a cada documento da entrada do estágio, os acumuladores **$avg**, **$max**, **$min**, **$stdDevPop**, **$stdDevSamp** e **$sum** devem ser usados no formato:

```
{ operador: <expression> }
```

Onde a expressão deve ser resolvida em um array para aplicar o operador sobre os elementos numéricos do array. Se a expressão for resolvida em um elemento numérico, o cálculo será aplicado a esse único elemento.

```
> db.alunos.aggregate(
    {$project: {_id: 0,
                nome: 1,
                notas: 1,
                media: { $avg: "$notas"}}}
  )
```

Estágios de Agregação: $project

```
{ "nome" : "Joao", "notas" : [ 8, 9, 5 ],
"media" : 7.333333333333333 }
{ "nome" : "Marcia", "notas" : [ 7, 6, 9 ],
"media" : 7.333333333333333 }
{ "nome" : "Paulo", "notas" : [ 5, 4, 5 ],
"media" : 4.666666666666667 }
{ "nome" : "Ana", "notas" : [ 5, 8, 9 ],
"media" : 7.333333333333333 }
{ "nome" : "Maria", "notas" : [ 8, 10, 7 ],
"media" : 8.333333333333334 }
{ "nome" : "Jose", "notas" : [ 10, 10, 9 ],
"media" : 9.666666666666666 }
```

Estágios de Agregação: $project

Ou no formato:

```
{operador:[<expression1>,<expression2>...]}
```

Onde as expressões devem ser resolvidas em elementos para aplicar o operador sobre os resultados das expressões que forem elementos numéricos. Se alguma expressão for resolvida em um array, esse array será tratado como valor não numérico e não será utilizado no cálculo, mesmo que os elementos do array sejam numéricos.

```
> db.produtos.aggregate(
    {$project: {_id: 0,
                item: 1,
                quantidade: 1,
                minimo: 1,
                media:{$avg:
["$quantidade","$minimo"]}}} )
```

Estágios de Agregação: $project

```
{  "item"  :  "caderno",  "quantidade"  :  20,
"media" : 20 }
{ "item" : "lapis", "quantidade" : 50,
"minimo" : 30, "media" : 40 }
{ "item" : "cartao", "quantidade" : 15,
"minimo" : 5, "media" : 10 }
{ "item" : "borracha", "quantidade" : 25,
"minimo" : 10, "media" : 17.5 }
{ "item" : "envelope", "quantidade" : 20,
"minimo" : 10, "media" : 15 }
{ "item" : "selos", "quantidade" : 30,
"minimo" : null, "media" : 30 }
```

Operadores

Além dos acumuladores, o MongoDB possui vários operadores que podem ser usados em expressões nos estágios de agregações. Existem operadores aritméticos, comparativos, lógicos, condicionais, para arrays, para datas, para strings, para conversão de tipo e outros.

Deve ser verificada a documentação do MongoDB sobre os operadores existentes além dos exemplos vistos a seguir.

Operador $add

$add retorna a soma das expressões.

```
{ $add: [ <expression1>, <expression2>, ... ]
}
> db.catalogo.aggregate(
    {$match: {preco: {$gte:2500.00}}},
    {$project: {_id: 0,
                item: 1,
                preco: 1,
                taxas: 1,
                cheio: { $add:
["$preco","$taxas"]}}}
  )
{ "item" : "TV", "preco" : 2500, "taxas" :
520.5, "cheio" : 3020.5 }
{ "item" : "Notebook", "preco" : 2900,
"taxas" : 320, "cheio" : 3220 }
```

Operador $subtract

$subtract retorna a subtração de duas expressões.

```
{ $push: <expression> }
```

```
> db.produtos.aggregate(
    {$project: {_id: 0,
                item: 1,
                qtd: "$quantidade",
                min: "$minimo",
                dif: {$subtract:
["$quantidade", "$minimo"]}}}
  )
```

Operador $subtract

```
{ "item" : "envelope", "qtd" : 20, "min" : 10, "dif" : 10 }
{ "item" : "selos", "qtd" : 30, "min" : null, "dif" : null }
{ "item" : "lapis", "qtd" : 50, "min" : 30, "dif" : 20 }
{ "item" : "cartao", "qtd" : 15, "min" : 5, "dif" : 10 }
{ "item" : "caderno", "qtd" : 20, "dif" : null }
{ "item" : "borracha", "qtd" : 25, "min" : 10, "dif" : 15 }
```

Operador $mutiply

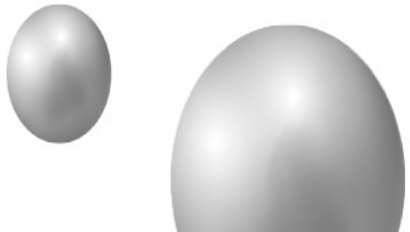
$multiply retorna a soma das expressões.

```
{ $multiply: [ <expression1>,
<expression2>, ... ] }
```

```
> db.catalogo.aggregate(
    {$match: {tipo: "Eletronicos"}},
    {$project: {_id: 0,
                item: 1,
                cheio: { $add:
["$preco","$taxas"]},
                desconto: 1,
                vl_desc: { $multiply: [{$add:
["$preco", "$taxas"]}, "$desconto", 0.01]}}}
  )
```

Operador $mutiply

```
{ "item" : "DVD Player", "desconto" : 10,
"cheio" : 280, "vl_desc" : 28 }
{ "item" : "SoundBar", "cheio" : 1200,
"vl_desc" : null }
{ "item" : "TV", "desconto" : 10, "cheio" :
3020.5, "vl_desc" : 302.05 }
```



Operador $divide

$divide retorna a divisão de duas expressões.

```
{ $divide: [ <expression1>, <expression2> ] }
```

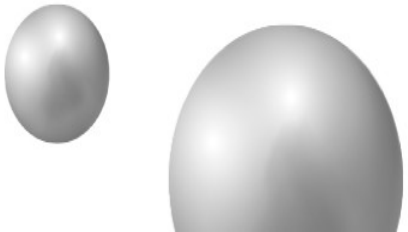
```
> db.catalogo.aggregate(
    {$match: {preco: {$lt:550.00}}},
    {$project: {_id: 0,
                item: 1,
                preco: 1,
                taxas: 1,
                perc: { $divide:
["$taxas","$preco"]}}}
  )
{ "item" : "DVD Player", "preco" : 250,
"taxas" : 30, "perc" : 0.12 }
{ "item" : "Impressora", "preco" : 520,
"taxas" : 60, "perc" : 0.11538461538461539 }
```

Operador $ceil

$ceil retorna o menor inteiro maior ou igual ao valor da expressão, a expressão deve retornar um valor numérico.

```
{ $ceil: <number> }
```

```
> db.produtos.aggregate(
    {$group: {_id: null,
              media: {$avg: "$quantidade"}}},
    { $project: {media: 1,
                 ceil: { $ceil:
"$media" } } }
  )
{ "_id" : null, "media" : 26.666666666666668,
"ceil" : 27 }
```



Operador $floor

$floor retorna o maior inteiro menor ou igual ao valor da expressão, a expressão deve retornar um valor numérico.

```
{ $floor: <number> }
```

```
> db.produtos.aggregate(
    {$group: {_id: null,
              media: {$avg: "$quantidade"}}},
    { $project: {media: 1,
                 floor: { $floor:
"$media" } } }
  )
{ "_id" : null, "media" : 26.666666666666668,
"floor" : 26 }
```

Operador $cmp

$cmp compara dois valores e retorna -1 se o primeiro for menor que o segundo, 0 se forem iguais ou 1 se o primeiro for maior que o segundo.

```
{ $cmp: [ <expression1>, <expression2> ] }
```

```
> db.produtos.aggregate(
    {$project: {_id: 0,
                item: 1,
                quantidade: 1,
                cmpTo25: { $cmp:
[ "$quantidade", 25 ] }}}
  )
```

Operador $cmp

```
{  "item"  :  "cartao",  "quantidade"  :  15,
"cmpTo25" : -1 }
{ "item" : "envelope", "quantidade" : 20,
"cmpTo25" : -1 }
{ "item" : "selos", "quantidade" : 30,
"cmpTo25" : 1 }
{ "item" : "lapis", "quantidade" : 50,
"cmpTo25" : 1 }
{ "item" : "caderno", "quantidade" : 20,
"cmpTo25" : -1 }
{ "item" : "borracha", "quantidade" : 25,
"cmpTo25" : 0 }
```

Operador $eq

$eq compara se duas expressões são iguais ou não.

```
{ $eq: [ <expression1>, <expression2> ] }
```

Além do operador **$eq** existem também os operadores **$gt**, **$gte**, **$lt**, **$lte** e **$ne** para comparar expressões.

```
> db.travelers.aggregate(
    {$project: {_id: 0,
                name: 1,
                isEliot: { $eq: [ "$name",
"Eliot" ] }}}
  )
{ "name" : "Dev", "isEliot" : false }
{ "name" : "Eliot", "isEliot" : true }
{ "name" : "Jeff", "isEliot" : false }
```

Operador $and

$and avalia as expressões e retorna verdadeiro se todas as expressões forem verdadeiras.

```
{ $and: [ <expression1>, <expression2>, ... ]
}
```

Além do operador **$and** existem também os operadores lõgicos **$or** e **$not**.

```
> db.produtos.aggregate(
    {$project: {_id: 0,
                item: 1,
                quantidade: 1,
                dentro: { $and: [ { $gte:
[ "$quantidade", 20 ] }, { $lte:
[ "$quantidade", 30 ] } ] }}}
  )
```

Operador $and

```
{  "item"  :  "cartao",  "quantidade"  :  15,
"dentro" : false }
{ "item" : "envelope", "quantidade" : 20,
"dentro" : true }
{ "item" : "selos", "quantidade" : 30,
"dentro" : true }
{ "item" : "lapis", "quantidade" : 50,
"dentro" : false }
{ "item" : "caderno", "quantidade" : 20,
"dentro" : true }
{ "item" : "borracha", "quantidade" : 25,
"dentro" : true }
```

Operador $arrayElemAt

$arrayElemAt retorna o elemento de um array de uma determada posição.

```
{ $arrayElemAt: [ <array>, <idx> ] }
```

```
> db.alunos.aggregate(
    { $project: {_id: 0,
                 nome: 1,
                 primeiro: { $arrayElemAt:
["$notas",0] } } }
  )
{ "nome" : "Joao", "primeiro" : 8 }
{ "nome" : "Jose", "primeiro" : 10 }
{ "nome" : "Maria", "primeiro" : 8 }
{ "nome" : "Marcia", "primeiro" : 7 }
{ "nome" : "Paulo", "primeiro" : 5 }
{ "nome" : "Ana", "primeiro" : 5 }
```

Operador $arrayElemAt

Se a posição for negativa, conta a posição a partir do final do array.

```
{ $arrayElemAt: [ <array>, <idx> ] }
```

```
> db.alunos.aggregate(
    { $project: {_id: 0,
                 nome: 1,
                 ultimo: { $arrayElemAt:
["$notas",-1] } } }
  )
{ "nome" : "Joao", "ultimo" : 5 }
{ "nome" : "Jose", "ultimo" : 9 }
{ "nome" : "Maria", "ultimo" : 7 }
{ "nome" : "Marcia", "ultimo" : 9 }
{ "nome" : "Paulo", "ultimo" : 5 }
{ "nome" : "Ana", "ultimo" : 9 }
```

Operador $indexOfArray

$indexOfArray retorna a posição d0 primeiro elemento do array igual a uma expressão. Se for especificado <start> e <end>, a pesquisa será feita apenas entre as posições a partir de <start> até antes de <end>.

```
{ $indexOfArray: [ <array expression>,
<search expression>, <start>, <end> ] }
```

```
> db.alunos.aggregate(
    { $project: {_id: 0,
                 nome: 1,
                 posicao: { $indexOfArray:
["$notas",5,0,2] } } }
  )
```

Operador $indexOfArray

```
{ "nome" : "Joao", "posicao" : -1 }
{ "nome" : "Jose", "posicao" : -1 }
{ "nome" : "Maria", "posicao" : -1 }
{ "nome" : "Marcia", "posicao" : -1 }
{ "nome" : "Paulo", "posicao" : 0 }
{ "nome" : "Ana", "posicao" : 0 }
```

Operador $arrayToObject

$arrayToObject converte um array em um documento único. Os elementos do arraretorna a divisão de duas expressões.

```
{ $arrayToObject: <expression> }
```

Os elementos do array (um para cada campo a criar) devem ser array com dois elementos: nome e valor do campo.

```
[ [<nome1>,<valor1>], [<nome2>,<valor2], ...]
```

Ou documentos (um para cada campo a criar) com os campos **k** para o nome do campo e **v** para o valor do campo.

```
[{"k": <nome1>,"v": <valor1>}, {"k": <nome2>,
"v": <valor2>}, ... ]
```

Se houver campo com nome duplicado, depende da versão ou do driver para decidir qual valor será usado.

Operador $arrayToObject

```
> db.catalogo.aggregate(
    {$match: {tipo: {$ne:"Informatica"}}},
    {$project: {_id: 0,
                item: 1,
                dimensoes: { $arrayToObject:
"$dimensoes" }}}
  )
```

Operador $arrayToObject

```
{ "item" : "Guarda Roupa", "dimensoes" :
{ "L" : 274, "A" : 235, "P" : 55.5, "um" :
"cm" } }
{ "item" : "DVD Player", "dimensoes" :
{ "A" : 16, "L" : 63, "P" : 46, "um" : "cm" }
}
{ "item" : "Sofa", "dimensoes" : { "A" :
0.98, "L" : 2.1, "P" : 1.18, "um" : "m" } }
{ "item" : "Mesa", "dimensoes" : { "L" : 80,
"A" : 76.5, "P" : 140, "um" : "cm" } }
{ "item" : "SoundBar", "dimensoes" : { "L" :
85.5, "A" : 14.5, "P" : 12.5, "um" : "cm" } }
{ "item" : "TV", "dimensoes" : { "L" :
1116.8, "A" : 664.2, "P" : 59.9, "um" :
"mm" } }
```

Operador $objectToArray

$objectToArray converte um documento para um array que tem como elementos, um documento para cada campo do documento original, com os campos k para o nome do campo e v para o valor do campo.

```
{ $objectToArray: <object> }
```

```
> db.editora.aggregate(
    {$project: {_id: 0,
                nome: 1,
                sede: { $objectToArray:
"$sede" }}}
  )
```

Operador $objectToArray

```
{ "nome" : "O'Reilly Media", "sede" : [ { "k"
: "pais", "v" : "Estados Unidos" }, { "k" :
"estado", "v" : "CA" }, { "k" : "cidade", "v"
: "Sebastopol" } ] }
{ "nome" : "Novatec Editora Ltda", "sede" : [
{ "k" : "pais", "v" : "Brasil" }, { "k" :
"estado", "v" : "SP" }, { "k" : "cidade", "v"
: "Sao Paulo" } ] }
{ "nome" : "Editora Ciencia Moderna",
"sede" : [ { "k" : "pais", "v" : "Brasil" },
{ "k" : "estado", "v" : "RJ" }, { "k" :
"cidade", "v" : "Rio de Janeiro" } ] }
```

Operador $filter

$filter filtra os elementos de um array segundo uma condição.

```
{ $filter: { input: <array>, as: <string>,
cond: <expression> } }
```

```
> db.acervo.aggregate(
    {$project: {_id: 0,
                editora: 1,
                Livros:{$filter:{input:
"$livros",
                                 as: "item",
                                 cond: { $eq:
[ "$$item.ano", "2010" ] }}}}}
  )
```

Operador $filter

```
{   "editora"   :   "Novatec   Editora   Ltda",
"livros" : [ ] }
{ "editora" : "Editora Ciencia Moderna",
"livros" : [ { "titulo" : "Programacao em
Linguagem C", "isbn" : "8573939494", "ano" :
"2010", "edicao" : "1" } ] }
{ "editora" : "O'Reilly Media", "livros" :
[ { "titulo" : "Programming With QT",
"isbn" : "9781449390938", "ano" : "2010",
"edicao" : "2" } ] }
```

Operador $size

$size retorna o numero de elementos de um array.

```
{ $size: <expression> }
```

```
> db.vendas.aggregate(
    {$project: {num_produtos: {$size:
"$produtos"}}}
  )
{ "_id" : "001", "num_produtos" : 2 }
{ "_id" : "004", "num_produtos" : 3 }
{ "_id" : "002", "num_produtos" : 3 }
{ "_id" : "003", "num_produtos" : 3 }
{ "_id" : "005", "num_produtos" : 5 }
{ "_id" : "006", "num_produtos" : 6 }
```

Operador $slice

$slice retorna um subconjunto de um array.

```
{ $slice: [ <array>, <position>, <n> ] }
```

Se não for especificada uma posição, retornar os elementos a partir do início do array.

```
> db.alunos.aggregate(
    { $project: {_id: 0,
                 nome: 1,
                 subset: { $slice:
["$notas",2] }}} )
{ "nome" : "Maria", "subset" : [ 8, 10 ] }
{ "nome" : "Joao", "subset" : [ 8, 9 ] }
{ "nome" : "Paulo", "subset" : [ 5, 4 ] }
{ "nome" : "Ana", "subset" : [ 5, 8 ] }
{ "nome" : "Jose", "subset" : [ 10, 10 ] }
{ "nome" : "Marcia", "subset" : [ 7, 6 ] }
```

Operador $slice

Se for especificada uma posição, negativa, retorna os elementos a partir do final do array.

```
> db.alunos.aggregate(
    { $project: {_id: 0,
                 nome: 1,
                 subset: { $slice:
["$notas",-2,2] }}}
  )
{ "nome" : "Maria", "subset" : [ 10, 7 ] }
{ "nome" : "Joao", "subset" : [ 9, 5 ] }
{ "nome" : "Paulo", "subset" : [ 4, 5 ] }
{ "nome" : "Ana", "subset" : [ 8, 9 ] }
{ "nome" : "Jose", "subset" : [ 10, 9 ] }
{ "nome" : "Marcia", "subset" : [ 6, 9 ] }
```

Operador $map

$map associa, mapeia, cada elemento do vetor com o resultado de uma expressão.

```
{ $map: { input: <expression>, as: <string>,
in: <expression> } }
```

```
> db.acervo.aggregate(
    {$project: {isbn: {$map: {input:
"$livros",
                                as: "livro",
                                in: "$
$livro.isbn"}}}}
  )
{ "_id" : "cm", "isbn" : [ "8573939494" ] }
{ "_id" : "oreilly", "isbn" :
[ "9781449390938", "9780596552367" ] }
{ "_id" : "novatec", "isbn" : [ "8575224220",
"8575224042" ] }
```

Operador $reduce

$reduce aplica uma expressão a cada elemento de um array e combina os resultados em um valor único.

```
{
    $reduce: {
        input: <array>,
        initialValue: <expression>,
        in: <expression>
    }
}
```

Operador $reduce

```
> db.acervo.aggregate(
    {$project:
      {"livros":
        {$reduce: {
          input: "$livros",
          initialValue:
{isbn:"Codigos:",total:0},
          in: {isbn:
                {$concat: ["$$value.isbn",
                           {$cond: [{ $eq:
[ "$$value.isbn", "Codigos:" ] }," ",", "]},
                           "$$this.isbn"]},
               total: {$add : ["$
$value.total",1] }}}}}}
  )
```

Operador $reduce

```
{ "_id" : "cm", "livros" : { "isbn" :
"Codigos: 8573939494", "total" : 1 } }
{ "_id" : "oreilly", "livros" : { "isbn" :
"Codigos: 9781449390938, 9780596552367",
"total" : 2 } }
{ "_id" : "novatec", "livros" : { "isbn" :
"Codigos: 8575224220, 8575224042", "total" :
2 } }
```

Operador $cond

$cond retorna um valor se uma expressão for verdadeira ou retorna um valor alternativo se a expressão for falsa.

```
{ $cond: { if: <boolean-expression>, then: <true-case>, else: <false-case> } }
```

Ou:

```
{ $cond: [ <boolean-expression>, <true-case>, <false-case> ] }
```

```
> db.alunos.aggregate(
    {$project: {_id: 0,
                nome: 1,
                media: { $avg: "$notas"},
                situacao: {$cond: [{$gte: [{$avg: "$notas"}, 7 ]}, "aprovado", "reprovado" ]}}}
  )
```

Operador $cond

```
{    "nome"    :    "Maria",    "media"    :
8.333333333333334, "situacao" : "aprovado" }
{ "nome" : "Joao", "media" :
7.333333333333333, "situacao" : "aprovado" }
{ "nome" : "Paulo", "media" :
4.666666666666667, "situacao" : "reprovado" }
{ "nome" : "Ana", "media" :
7.333333333333333, "situacao" : "aprovado" }
{ "nome" : "Jose", "media" :
9.666666666666666, "situacao" : "aprovado" }
{ "nome" : "Marcia", "media" :
7.333333333333333, "situacao" : "aprovado" }
```

Operador $ifNull

$ifNull retorna o valor de uma expressão se não for nula ou retorna um valor alternativo se a expressão for nula.

```
{ $ifNull: [ <expression>, <replacement-expression-if-null> ] }
```

```
> db.produtos.aggregate(
    {$project: {_id: 0,
                item: 1,
                qtd: "$quantidade",
                min: "$minimo",
                dif: {$subtract:
["$quantidade", {$ifNull: ["$minimo", 0]}]}}}
  )
```

Operador $ifNull

```
{ "item" : "cartao", "qtd" : 15, "min" : 5,
"dif" : 10 }
{ "item" : "envelope", "qtd" : 20, "min" :
10, "dif" : 10 }
{ "item" : "selos", "qtd" : 30, "min" : null,
"dif" : 30 }
{ "item" : "lapis", "qtd" : 50, "min" : 30,
"dif" : 20 }
{ "item" : "caderno", "qtd" : 20, "dif" :
20 }
{ "item" : "borracha", "qtd" : 25, "min" :
10, "dif" : 15 }
```

Operador $switch

$switch retorna um valor dependendo de uma série de condições.

```
{ $switch: {
    branches: [
        { case: <expression>, then:
<expression> },
        { case: <expression>, then:
<expression> },
        ...
    ],
    default: <expression>
} }
```

Operador $switch

```
> db.alunos.aggregate(
    {$project: {_id: 0,
                nome: 1,
                media: { $avg: "$notas"},
                situacao: {$switch:
                              {branches: [
                                          {case:
{$gte: [{$avg: "$notas"}, 9]}, then:
"excelente"},
                                          {case:
{$gte: [{$avg: "$notas"}, 7]}, then: "bom"},
                                          ],
                                          default:
"reprovado" }}}}
  )
```

Operador $switch

```
{ "nome" : "Maria", "media" :
8.333333333333334, "situacao" : "bom" }
{ "nome" : "Joao", "media" :
7.333333333333333, "situacao" : "bom" }
{ "nome" : "Paulo", "media" :
4.666666666666667, "situacao" : "reprovado" }
{ "nome" : "Ana", "media" :
7.333333333333333, "situacao" : "bom" }
{ "nome" : "Jose", "media" :
9.666666666666666, "situacao" : "excelente" }
{ "nome" : "Marcia", "media" :
7.333333333333333, "situacao" : "bom" }
```

Operador $dateFromString

$dateFromString converte uma string para date/time.

```
{ $dateFromString: {
      'dateString': <dateStringExpression>,
      'timezone': <tzExpression>
  }
}
```

```
> db.vendas.aggregate(
    {$project: {_id: 0,
                data: 1,
                date: {$dateFromString:
{dateString:"$data"}}}}
  )
```

Operador $dateFromString

```
{ "data" : "2020-02-14", "date" :
ISODate("2020-02-14T00:00:00Z") }
{ "data" : "2020-02-22", "date" :
ISODate("2020-02-22T00:00:00Z") }
{ "data" : "2020-02-18", "date" :
ISODate("2020-02-18T00:00:00Z") }
{ "data" : "2020-02-18", "date" :
ISODate("2020-02-18T00:00:00Z") }
{ "data" : "2020-03-02", "date" :
ISODate("2020-03-02T00:00:00Z") }
{ "data" : "2020-03-06", "date" :
ISODate("2020-03-06T00:00:00Z") }
```

Operador $dayOfMonth

$dayOfMonth retorna o dia do mês de uma data.

```
{ $dayOfMonth: <dateExpression> }
```

```
> db.vendas.aggregate(
    {$project: {_id: 0,
                data: 1,
                dia: {$dayOfMonth:
{$dateFromString: {dateString:"$data"}}}}}
  )
{ "data" : "2020-02-14", "dia" : 14 }
{ "data" : "2020-02-22", "dia" : 22 }
{ "data" : "2020-02-18", "dia" : 18 }
{ "data" : "2020-02-18", "dia" : 18 }
{ "data" : "2020-03-02", "dia" : 2 }
{ "data" : "2020-03-06", "dia" : 6 }
```

Operador $month

$month retorna o número do mês de uma data.

```
{ $month: <dateExpression> }
```

```
> db.vendas.aggregate(
    {$project: {_id: 0,
                data: 1,
                dia: {$month:
{$dateFromString: {dateString:"$data"}}}}}
  )
{ "data" : "2020-02-14", "dia" : 2 }
{ "data" : "2020-02-22", "dia" : 2 }
{ "data" : "2020-02-18", "dia" : 2 }
{ "data" : "2020-02-18", "dia" : 2 }
{ "data" : "2020-03-02", "dia" : 3 }
{ "data" : "2020-03-06", "dia" : 3 }
```

Operador $year

$year retorna o ano de uma data.

```
{ $year: <dateExpression> }
```

```
> db.vendas.aggregate(
    {$project: {_id: 0,
                data: 1,
                dia: {$year:
{$dateFromString: {dateString:"$data"}}}}}
  )
{ "data" : "2020-02-14", "dia" : 2020 }
{ "data" : "2020-02-22", "dia" : 2020 }
{ "data" : "2020-02-18", "dia" : 2020 }
{ "data" : "2020-02-18", "dia" : 2020 }
{ "data" : "2020-03-02", "dia" : 2020 }
{ "data" : "2020-03-06", "dia" : 2020 }
```

Operador $let

$let define variáveis para uso em uma expressão.

```
{
  $let:
     {
       vars: { <var1>: <expression>, ... },
       in: <expression>
     }
}
```

As variáveis definidas em vars só podem ser utilizadas na expressão em in.

Operador $let

```
> db.catalogo.aggregate(
    {$match: {tipo: "Eletronicos"}},
    {$project:
      {_id: 0,
       item: 1,
       desconto: 1,
       vl_desc:
         {$let: {vars: {cheio: {$add:
["$preco","$taxas"]}},
          in: { $multiply: ["$$cheio",
"$desconto", 0.01]}}}}} )
{ "item" : "SoundBar", "vl_desc" : null }
{ "item" : "TV", "desconto" : 10, "vl_desc" :
302.05 }
{ "item" : "DVD Player", "desconto" : 10,
"vl_desc" : 28 }
```

Exercícios

1) Relacionar o id da venda, nome do produto e valor total do produto para os produtos comprados pela Laura, ordenados pelo nome do produto

```
{ "_id" : "001", "produto" : "Caderno",
"total" : 21 }
{ "_id" : "005", "produto" : "Caderno",
"total" : 42.900000000000006 }
{ "_id" : "005", "produto" : "Caneta",
"total" : 12.8 }
{ "_id" : "005", "produto" : "Corretivo",
"total" : 5 }
{ "_id" : "005", "produto" : "Fichario",
"total" : 50.2 }
{ "_id" : "001", "produto" : "Lapis", "total"
: 3.4499999999999997 }
```

Exercícios

```
{  "_id"  :  "005",  "produto"  :  "Post-it",
"total" : 56.4 }
```

Exercícios

2) Relacionar o nome do produto dos produtos comprados pela Laura e o total gasto com o produto, em ordem decrescente do total gasto

```
{ "_id" : "Caderno", "total" :
63.900000000000006 }
{ "_id" : "Post-it", "total" : 56.4 }
{ "_id" : "Fichario", "total" : 50.2 }
{ "_id" : "Caneta", "total" : 12.8 }
{ "_id" : "Corretivo", "total" : 5 }
{ "_id" : "Lapis", "total" :
3.4499999999999997 }
```

Exercícios

3) Relacionar o id da venda e o valor total da venda

```
{ "_id" : "001", "total" : 24.45 }
{ "_id" : "002", "total" : 84.85 }
{ "_id" : "003", "total" : 17.6 }
{ "_id" : "004", "total" : 70.9 }
{ "_id" : "005", "total" : 167.3 }
{ "_id" : "006", "total" : 252.9 }
```

Exercícios

4) Relacionar o nome do cliente e o valor total gasto pelo cliente, ordenado pelo nome do cliente

```
{ "_id" : "Henrique", "total" : 17.6 }
{ "_id" : "Laura", "total" : 191.75 }
{ "_id" : "Marta", "total" : 70.9 }
{ "_id" : "Paulo", "total" : 84.85 }
{ "_id" : "Renata", "total" : 252.9 }
```

Exercícios

5) Relacionar o mês e o valor total das vendas do mês para as compras do ano de 2020

```
{ "_id" : 2, "total" : 197.8 }
{ "_id" : 3, "total" : 420.20000000000005 }
```

Exercícios

6) Relacionar o id da venda, nome do cliente e dados do produto “Caneta” para as vendas que tenham o produto “Caneta”

```
{ "_id" : "002", "cliente" : "Paulo",
"produtos" : [ { "nome" : "Caneta",
"quantidade" : 2, "preco" : 3.2 } ] }
{ "_id" : "005", "cliente" : "Laura",
"produtos" : [ { "nome" : "Caneta",
"quantidade" : 4, "preco" : 3.2 } ] }
{ "_id" : "006", "cliente" : "Renata",
"produtos" : [ { "nome" : "Caneta",
"quantidade" : 6, "preco" : 3.1 } ] }
```

Exercícios

7) Relacionar o nome, volume, unidade de medida das dimensões do item e o volume em m³ dos itens do catálogo


```
{ "item" : "DVD Player", "volume" : 46368,
"unidade" : "cm", "convertido" : 0.046368 }
{ "item" : "Guarda Roupa", "volume" :
3573645, "unidade" : "cm", "convertido" :
3.573645 }
{ "item" : "Impressora", "volume" : 19896.8,
"unidade" : "cm", "convertido" : 0.0198968 }
{ "item" : "Mesa", "volume" : 856800,
"unidade" : "cm", "convertido" : 0.8568 }
{ "item" : "Monitor", "volume" : 3954600,
"unidade" : "mm", "convertido" : 0.0039546 }
{ "item" : "Notebook", "volume" : 1950.996,
"unidade" : "cm", "convertido" :
0.0019509960000000002 }
```

Exercícios


```
{    "item"    :    "Sofa",    "volume"    :
2.4284399999999997,    "unidade"    :    "m",
"convertido" : 2.4284399999999997 }
{ "item" : "SoundBar", "volume" : 15496.875,
"unidade" : "cm", "convertido" :
0.015496875 }
{ "item" : "TV", "volume" : 44432535.744,
"unidade" : "mm", "convertido" :
0.044432535744 }
```

Exercícios

8) Relacionar o nome do item, o preço cheio (preço+taxas) e o preço final (preço cheio menos o desconto) para os itens do catálogo

```
{ "item" : "DVD Player", "cheio" : 280,
"final" : 252 }
{ "item" : "Guarda Roupa", "cheio" : 2600,
"final" : 2210 }
{ "item" : "Impressora", "cheio" : 580,
"final" : 536.5 }
{ "item" : "Mesa", "cheio" : 1340, "final" :
1273 }
{ "item" : "Monitor", "cheio" : 635,
"final" : 584.2 }
{ "item" : "Notebook", "cheio" : 3220,
"final" : 3220 }
```

Exercícios

```
{ "item" : "Sofa", "cheio" : 2010, "final" : 1809 }
{ "item" : "SoundBar", "cheio" : 1200, "final" : 1200 }
{ "item" : "TV", "cheio" : 3020.5, "final" : 2718.4500000000003 }
```