
UNITAU
Universidade de Taubaté

MongoDB Agregação

Estágios de Agregação: $addFields

$addFields acrescenta novos campos nos documentos da entrada do estágio.

```
{ $addFields: { <newField>: <expression>, ...
} }
```

```
> db.alunos.aggregate(
    {$addFields:
      {total:
        {$sum: "$notas"},
         media: {$trunc: {$avg: "$notas"}}}}
  )
```

Estágios de Agregação: $addFields


```
{ "_id" : 1, "nome" : "Joao", "notas" : [ 8, 9, 5 ], "total" : 22, "media" : 7 }
{ "_id" : 2, "nome" : "Maria", "notas" : [ 8, 10, 7 ], "total" : 25, "media" : 8 }
{ "_id" : 3, "nome" : "Jose", "notas" : [ 10, 10, 9 ], "total" : 29, "media" : 9 }
{ "_id" : 4, "nome" : "Marcia", "notas" : [ 7, 6, 9 ], "total" : 22, "media" : 7 }
{ "_id" : 5, "nome" : "Paulo", "notas" : [ 5, 4, 5 ], "total" : 14, "media" : 4 }
{ "_id" : 6, "nome" : "Ana", "notas" : [ 5, 8, 9 ], "total" : 22, "media" : 7 }
```

Estágios de Agregação: $addFields

$addFields pode ser utilizado para adicionar campos a documentos embutidos usando a notação de pontos.

```
> db.editora.aggregate(
    {$addFields: {"sede.telefone": "n/c"}}
  )
```

Estágios de Agregação: $addFields

```
{ "_id" : "oreilly", "nome" : "O'Reilly Media", "sede" : { "pais" : "Estados Unidos", "estado" : "CA", "cidade" : "Sebastopol", "telefone" : "n/c" } }
{ "_id" : "novatec", "nome" : "Novatec Editora Ltda", "sede" : { "pais" : "Brasil", "estado" : "SP", "cidade" : "Sao Paulo", "telefone" : "n/c" } }
{ "_id" : "cm", "nome" : "Editora Ciencia Moderna", "sede" : { "pais" : "Brasil", "estado" : "RJ", "cidade" : "Rio de Janeiro", "telefone" : "n/c" } }
```

Estágios de Agregação: $addFields

$addFields pode sobrepor o valor de um campo já existente.

```
> db.produtos.aggregate( {$addFields:
{minimo: 45}} )
{ "_id" : 1, "item" : "caderno", "quantidade"
: 20, "minimo" : 45 }
{ "_id" : 2, "item" : "cartao",
"quantidade" : 15, "minimo" : 45 }
{ "_id" : 3, "item" : "envelope",
"quantidade" : 20, "minimo" : 45 }
{ "_id" : 4, "item" : "selos", "quantidade" :
30, "minimo" : 45 }
{ "_id" : 5, "item" : "lapis", "quantidade" :
50, "minimo" : 45 }
{ "_id" : 6, "item" : "borracha",
"quantidade" : 25, "minimo" : 45 }
```

Estágios de Agregação: $redact

$redact decompõe um array, gerando um documento para cada elemento do array com o valor do elemento substituindo o array.

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

A expressão deve ter como resultado uma das seguintes variáveis:

- **$$DESCEND** – mantém os campos do documento corrente, documentos embutidos e documentos embutidos em arrays serão avaliados pela expressão para determinar se serão acessados ou não.
- **$$PRUNE** – exclui todos os campos do documento corrente e em documentos embutidos.
- **$$KEEP** – mantém todos os campos do documento corrente e documentos embutidos.

Estágios de Agregação: $redact

```
> db.acervo.aggregate(
    {$redact:
      {$cond: {
        if: {$or: [{$eq: ["$editora",
"O'Reilly Media"]},
                   {$eq: ["$ano", "2010"]}]},
        then: "$$DESCEND",
        else: "$$PRUNE"}}}
  ).pretty()
```

Estágios de Agregação: $redact

```
{
        "_id" : "oreilly",
        "editora" : "O'Reilly Media",
        "livros" : [
                {
                        "titulo" :
"Programming With QT",
                        "isbn" :
"9781449390938",
                        "ano" : "2010",
                        "edicao" : "2"
                }
        ]
}
```

Estágios de Agregação: $redact

$replaceRoot substitui o documento superior pelo documento especificado. O documento especificado pode ser um documento embutido ou um novo documento criado no próprio estágio.

```
{ $replaceRoot: { newRoot:
<replacementDocument> } }
```

```
> db.editora.aggregate( {$replaceRoot:
{newRoot: "$sede"}} )
{ "pais" : "Estados Unidos", "estado" : "CA",
"cidade" : "Sebastopol" }
{ "pais" : "Brasil", "estado" : "SP",
"cidade" : "Sao Paulo" }
{ "pais" : "Brasil", "estado" : "RJ",
"cidade" : "Rio de Janeiro" }
```

Estágios de Agregação: $lookup

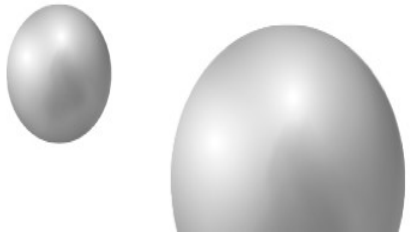
$lookup executa o LEFT OUTER JOIN dos documentos da entrada do estágio com uma coleção não fragmentada no mesmo banco de dados.

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

Estágios de Agregação: $lookup

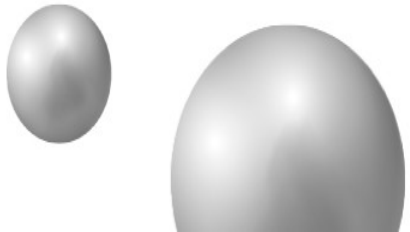
Ou

```
{$lookup:
      {
        from: <collection to join>,
        let: { <var_1>: <expression>, …,
<var_n>: <expression> },
        pipeline: [ <pipeline to execute on
the collection to join> ],
        as: <output array field>
      } }
```



Estágios de Agregação: $lookup

```
> db.livro.aggregate(
    {$match: {editora_id: "oreilly"}},
    {$lookup: {from: "editora",
               localField: "editora_id",
               foreignField: "_id",
               as: "editora"}}
  )
```



Estágios de Agregação: $lookup


```
{ "_id" : 3, "titulo" : "Programming With
QT", "editora_id" : "oreilly", "isbn" :
"9781449390938", "ano" : "2010", "edicao" :
"2", "editora" : [ { "_id" : "oreilly",
"nome" : "O'Reilly Media", "sede" :
{ "pais" : "Estados Unidos", "estado" : "CA",
"cidade" : "Sebastopol" } } ] }
{ "_id" : 4, "titulo" : "SQL Tuning",
"editora_id" : "oreilly", "isbn" :
"9780596552367", "ano" : "2009", "edicao" :
"1", "editora" : [ { "_id" : "oreilly",
"nome" : "O'Reilly Media", "sede" :
{ "pais" : "Estados Unidos", "estado" : "CA",
"cidade" : "Sebastopol" } } ] }
```

Estágios de Agregação: $lookup

```
> db.editora.aggregate(
    {$match: {_id: "oreilly"}},
    {$lookup: {from: "livro",
               localField: "_id",
               foreignField: "editora_id",
               as: "livros"}} )
```


```
{ "_id" : "oreilly", "nome" : "O'Reilly
Media", "sede" : { "pais" : "Estados Unidos",
"estado" : "CA", "cidade" : "Sebastopol" },
"livros" : [ { "_id" : 3, "titulo" :
"Programming With QT", "editora_id" :
"oreilly", "isbn" : "9781449390938", "ano" :
"2010", "edicao" : "2" }, { "_id" : 4,
"titulo" : "SQL Tuning", "editora_id" :
"oreilly", "isbn" : "9780596552367", "ano" :
"2009", "edicao" : "1" } ] }
```

Estágios de Agregação: $grahLookup

$graphLookup executa uma pesquisa recursiva em uma coleção não fragmentada no mesmo banco de dados.

```
{
   $graphLookup: {
      from: <collection>,
      startWith: <expression>,
      connectFromField: <string>,
      connectToField: <string>,
      as: <string>,
      maxDepth: <number>,
      depthField: <string>,
      restrictSearchWithMatch: <document>
   }
}
```

Estágios de Agregação: $grahLookup

```
> db.employees.insertMany( [
    {_id: 1, name: "Eliot"},
    {_id: 2, name: "Ron", reportsTo:
"Eliot"},
    {_id: 3, name: "Andrew", reportsTo:
"Eliot"},
    {_id: 4, name: "Asya", reportsTo: "Ron"},
    {_id: 5, name: "Dan", reportsTo:
"Andrew"}
    ] )
{ "acknowledged" : true, "insertedIds" : [ 1,
2, 3, 4, 5 ] }
```

Estágios de Agregação: $grahLookup

```
> db.employees.aggregate(
    {$sort: {_id: 1}},
    {$graphLookup:
      {from: "employees",
       startWith: "$reportsTo",
       connectFromField: "reportsTo",
       connectToField: "name",
       as: "reportingHierarchy"}}
  )
```

Estágios de Agregação: $grahLookup

```
{ "_id" : 1, "name" : "Eliot",
"reportingHierarchy" : [ ] }
{ "_id" : 2, "name" : "Ron", "reportsTo" :
"Eliot", "reportingHierarchy" : [ { "_id" :
1, "name" : "Eliot" } ] }
{ "_id" : 3, "name" : "Andrew", "reportsTo" :
"Eliot", "reportingHierarchy" : [ { "_id" :
1, "name" : "Eliot" } ] }
{ "_id" : 4, "name" : "Asya", "reportsTo" :
"Ron", "reportingHierarchy" : [ { "_id" : 1,
"name" : "Eliot" }, { "_id" : 2, "name" :
"Ron", "reportsTo" : "Eliot" } ] }
{ "_id" : 5, "name" : "Dan", "reportsTo" :
"Andrew", "reportingHierarchy" : [ { "_id" :
1, "name" : "Eliot" }, { "_id" : 3, "name" :
"Andrew", "reportsTo" : "Eliot" } ] }
```

Estágios de Agregação: $grahLookup

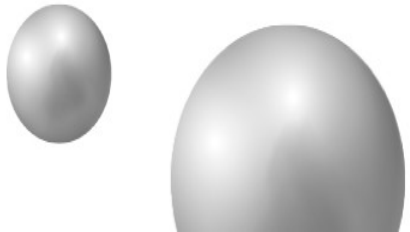
```
> db.airports.insertMany( [
    {_id: 0, airport: "JFK", connects:
["BOS", "ORD"]},
    {_id: 1, airport: "BOS", connects:
["JFK", "PWM"]},
    {_id: 2, airport: "ORD", connects:
["JFK"]},
    {_id: 3, airport: "PWM", connects:
["BOS", "LHR"]},
    {_id: 4, airport: "LHR", connects:
["PWM"]}
    ] )
{ "acknowledged" : true, "insertedIds" : [ 0,
1, 2, 3, 4 ] }
```

Estágios de Agregação: $grahLookup

```
> db.travelers.insertMany( [
    {_id: 1, name: "Dev", nearestAirport:
"JFK"},
    {_id: 2, name: "Eliot", nearestAirport:
"LHR"},
    {_id: 3, name: "Jeff", nearestAirport:
"BOS"}
    ] )
{ "acknowledged" : true, "insertedIds" : [ 1,
2, 3 ] }
```

Estágios de Agregação: $grahLookup

```
> db.travelers.aggregate(
    {$match: {name: "Dev"}},
    {$graphLookup:
      {from: "airports",
       startWith: "$nearestAirport",
       connectFromField: "connects",
       connectToField: "airport",
       depthField: "numConnections",
       as: "destinations"}}
  )
```



Estágios de Agregação: $grahLookup


```
{ "_id" : 1, "name" : "Dev", "nearestAirport"
: "JFK", "destinations" : [ { "_id" : 4,
"airport" : "LHR", "connects" : [ "PWM" ],
"numConnections" : NumberLong(3) }, { "_id" :
2, "airport" : "ORD", "connects" : [ "JFK" ],
"numConnections" : NumberLong(1) }, { "_id" :
0, "airport" : "JFK", "connects" : [ "BOS",
"ORD" ], "numConnections" : NumberLong(0) },
{ "_id" : 1, "airport" : "BOS", "connects" :
[ "JFK", "PWM" ], "numConnections" :
NumberLong(1) }, { "_id" : 3, "airport" :
"PWM", "connects" : [ "BOS", "LHR" ],
"numConnections" : NumberLong(2) } ] }
```

Estágios de Agregação: $grahLookup

```
> db.travelers.aggregate(
    {$match: {name: "Jeff"}},
    {$graphLookup:
      {from: "airports",
       startWith: "$nearestAirport",
       connectFromField: "connects",
       connectToField: "airport",
       depthField: "numConnections",
       as: "destinations"}}
  )
```

Estágios de Agregação: $grahLookup


```
{ "_id" : 3, "name" : "Jeff",
"nearestAirport" : "BOS", "destinations" :
[ { "_id" : 2, "airport" : "ORD",
"connects" : [ "JFK" ], "numConnections" :
NumberLong(2) }, { "_id" : 1, "airport" :
"BOS", "connects" : [ "JFK", "PWM" ],
"numConnections" : NumberLong(0) }, { "_id" :
0, "airport" : "JFK", "connects" : [ "BOS",
"ORD" ], "numConnections" : NumberLong(1) },
{ "_id" : 4, "airport" : "LHR", "connects" :
[ "PWM" ], "numConnections" :
NumberLong(2) }, { "_id" : 3, "airport" :
"PWM", "connects" : [ "BOS", "LHR" ],
"numConnections" : NumberLong(1) } ] }
```

Estágios de Agregação: $grahLookup

```
> db.travelers.aggregate(
    {$match: {name: "Eliot"}},
    {$graphLookup:
      {from: "airports",
       startWith: "$nearestAirport",
       connectFromField: "connects",
       connectToField: "airport",
       maxDepth: 3,
       depthField: "numConnections",
       as: "destinations"}}
  )
```

Estágios de Agregação: $grahLookup


```
{ "_id" : 2, "name" : "Eliot",
"nearestAirport" : "LHR", "destinations" :
[ { "_id" : 1, "airport" : "BOS",
"connects" : [ "JFK", "PWM" ],
"numConnections" : NumberLong(2) }, { "_id" :
3, "airport" : "PWM", "connects" : [ "BOS",
"LHR" ], "numConnections" : NumberLong(1) },
{ "_id" : 0, "airport" : "JFK", "connects" :
[ "BOS", "ORD" ], "numConnections" :
NumberLong(3) }, { "_id" : 4, "airport" :
"LHR", "connects" : [ "PWM" ],
"numConnections" : NumberLong(0) } ] }
```

Estágios de Agregação: $facet

$facet processa múltiplos pipelines de agregação e coloca o resultado de cada pipeline como um array no documento de saída do estágio. O estágio $facet permite a criação de agregações multidimensionais.

```
{ $facet:
   {
      <outputField1>: [ <stage1>,
<stage2>, ... ],
      <outputField2>: [ <stage1>,
<stage2>, ... ],
      ...

   }
}
```

Estágios de Agregação: $facet

Nos pipelines do estágio `$facet` não podem ser usados os seguintes estágios de agregação:

- `$facet`
- `$out`
- `$geoNear`
- `$indexStats`
- `$collStats`

Estágios de Agregação: $facet

```
> db.megasena.aggregate(
{$match:{sena_ganhadores:{$gt:4, $lt:8}}},
{$sort: {_id:1}},
{$facet: {
  relacao:[{$project:{qtd:"$sena_ganhadores",
            rateio:"$sena_rateio" }}],
  maiores:[{$group:{_id:"$sena_ganhadores",
            rateio:{$max:"$sena_rateio"}}}],
  menores:[{$group:{_id:"$sena_ganhadores",
            rateio:{$min:"$sena_rateio"}}}],
  ultimos:[{$group:{_id:"$sena_ganhadores",
            rateio:{$last:"$sena_rateio"}}}]}}
).pretty()
```

Estágios de Agregação: $facet


```
{
"relacao" : [
  {"_id":233,"qtd":5,"rateio":3196547.03},
  {"_id":1211,"qtd":7,"rateio":13217564.89},
  {"_id":1350,"qtd":5,"rateio":35523497.52},
  {"_id":1775,"qtd":6,"rateio":41088919.05},
  {"_id":1890,"qtd":6,"rateio":36824758.22}],
"maiores" : [
  {"_id":6,"rateio":41088919.05},
  {"_id":7,"rateio":13217564.89},
  {"_id":5,"rateio":35523497.52}],
"menores" : [
  {"_id":6,"rateio":36824758.22},
  {"_id":7,"rateio":13217564.89},
  {"_id":5,"rateio":3196547.03}],
```

Estágios de Agregação: $facet

```
"ultimos" : [
  {"_id" : 6, "rateio" : 36824758.22},
  {"_id" : 7, "rateio" : 13217564.89},
  {"_id" : 5, "rateio" : 35523497.52} ]
}
```

Estágios de Agregação: $out

$out armazena o resultado da agregação em uma coleção. Se a coleção destino já existe, a mesma será substituída pelo resultado da agregação. O estágio $out deve ser o último estágio da agregação.

```
{ $out: "<output-collection>" }
```

```
> db.megasena.aggregate(
    {$match: {sena_ganhadores: {$gt: 0}}},
    {$group: {_id : "$sena_ganhadores",
              sorteios: {$sum: 1},
              ganhadores: {$sum:
"$sena_ganhadores"}}},
    {$sort: {_id: 1}},
    {$out: "ganhadores_sena"}
  )
>
```

Estágios de Agregação: $out

```
> db.ganhadores_sena.find()
{"_id": 1, "sorteios": 358, "ganhadores":
358}
{"_id": 2, "sorteios": 87, "ganhadores": 174}
{"_id": 3, "sorteios": 23, "ganhadores": 69}
{"_id": 4, "sorteios": 11, "ganhadores": 44}
{"_id": 5, "sorteios": 2, "ganhadores": 10}
{"_id": 6, "sorteios": 2, "ganhadores": 12}
{"_id": 7, "sorteios": 1, "ganhadores": 7}
{"_id": 15, "sorteios": 1, "ganhadores": 15}
{"_id": 17, "sorteios": 1, "ganhadores": 17}
```

Estágios de Agregação: $geoNear

$geoNear ordena os documentos por proximidade espacial em relação a um ponto.

```
{ $geoNear: { <geoNear options> } }
```

Para utilizar o estágio $geoNear é necessário que a coleção tenha um índice geoespacial. O exemplo de utilização do estágio $geoNear será visto no tópico sobre índices geoespaciais.

Agregação em Banco de Dados

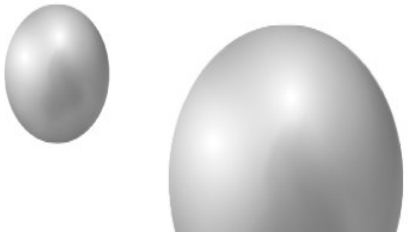
Também existem estágios de agregação para obter informações sobre as atividades e sessões do banco de dados. Essas agregações sobre o banco de dados devem ser verificadas na documentação do MongoDB e não podem ser utilizadas sobre coleções.

MapReduce

```
> db.notas.insertMany( [
{_id:1,nome:"Joao",disciplina:"ciencias",nota
:68},
{_id:2,nome:"Joao",disciplina:"matematica",no
ta:98},
{_id:3,nome:"Joao",disciplina:"portugues",not
a:77},
{_id:4,nome:"Andre",disciplina:"ciencias",not
a:67},
{_id:5,nome:"Andre",disciplina:"matematica",n
ota:87},
{_id:6,nome:"Andre",disciplina:"portugues",no
ta:89},
```

MapReduce

```
{_id:7,nome:"Ana",disciplina:"ciencias",nota:
67},
{_id:8,nome:"Ana",disciplina:"matematica",not
a:78},
{_id:9,nome:"Ana",disciplina:"portugues",nota
:90},
{_id:10,nome:"Paula",disciplina:"portugues",n
ota:85}
] )
{
        "acknowledged" : true,
        "insertedIds" :
[1,2,3,4,5,6,7,8,9,10]
}
```



MapReduce

Map-reduce é um paradigma de processamento de dados para condensar um grande volume de dados em resultados agregados. O método **db.collection.mapReduce** executa a operação de map-reduce sobre a coleção.

```
db.collection.mapReduce(
   <map>,
   <reduce>,
   { out: <collection>,
     query: <document>,
     sort: <document>,
     limit: <number>,
     finalize: <function>,
     scope: <document>,
     jsMode: <boolean>,
     verbose: <boolean>,
     bypassDocumentValidation: <boolean> } )
```

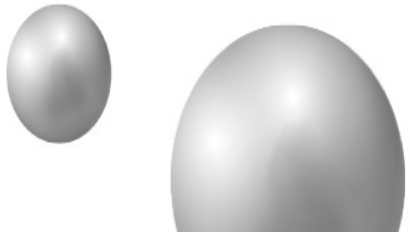
MapReduce

A operação de map-reduce executa uma agregação de duas fases sobre a coleção;

Map – aplica uma função em JavaScript sobre a coleção que associa, mapeia, cada documento da coleção em 0 ou mais documentos com um par chave/valor.

Reduce – aplica uma função JavaScript que “reduz” os valores associados a cada chave a um único valor. A redução não será aplicada para chaves que tenham apenas um único valor.

As funções de mapeamento e redução não devem acessar o banco de dados.



MapReduce

A função de redução pode ser aplicada mais de uma vez a uma chave e portanto deve ter algumas propriedades:

a) A função deve retornar um valor do mesmo tipo emitido pela função de mapeamento.

b) A função deve ser associativa:

```
reduce(key,[C,reduce(key,[A,B])]) ==
reduce(key,[C,A,B])
```

c) A função deve ser idempotente:

```
reduce(key,[reduce(key,values)]) ==
reduce(key,values)
```

d) A função deve ser comutativa:

```
reduce(key,[A,B]) == reduce(key,[B,A])
```

MapReduce

```
> var map = function()
{emit(this.nome,this.nota);};
>
> var reduce = function(nome,nota) {return
Array.sum(nota);};
>
```

MapReduce

```
> db.notas.mapReduce(
        map,
        reduce,
        { out: "notas_totais" }
    );
{
        "result" : "notas_totais",
        "timeMillis" : 1232,
        "counts" : {
                "input" : 10,
                "emit" : 10,
                "reduce" : 3,
                "output" : 4
        },
        "ok" : 1
}
```

MapReduce

```
> db.notas_totais.find()
{ "_id" : "Ana", "value" : 235 }
{ "_id" : "Andre", "value" : 243 }
{ "_id" : "Joao", "value" : 243 }
{ "_id" : "Paula", "value" : 85 }
```