

MongoDB: Framework de Agregação

Agenda

- MongoDB: Framework de agregação
 - Match
 - Project
 - Sort, skip e count
 - Sample
 - Group
 - Unwind
 - Lookup

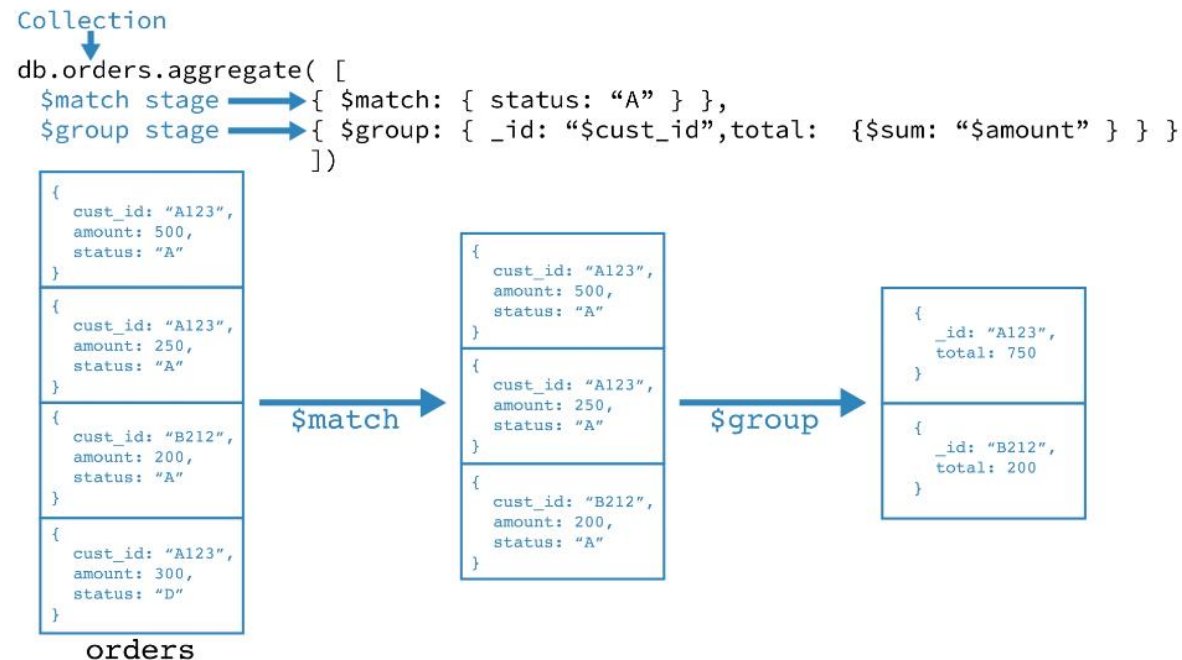
- No MongoDB, há duas formas básicas de **recuperação** de dados: através de consultas com o **find()** e através da análise utilizando a **aggregation framework** através da função **aggregate()**;
- **find()** permite filtrar resultados, fazer **transformações** básicas de documentos, **ordenar** os documentos, **limitar** o conjunto de resultados de documentos, etc;

Framework de agregação

- A **agregação** no MongoDB permite a transformação de dados de forma mais poderosa do que a utilização do comando `find()`;
- Através da utilização de múltiplas fases (**stages**) e expressões, é possível construir um "**pipeline**" de operações sobre os seus dados para realizar operações analíticas;
- Cada **stage** transfere a saída padrão para algum outro destino. A **saída** de um stage é **enviada** para outro **stage** para processamento posterior.

Framework de agregação

- Os documentos entram numa pipeline de várias **etapas** que os transforma num resultado **agregado**.



Framework de agregação

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

- **Primeira etapa:** a etapa: **\$match** filtra os documentos pelo campo de status e passa para a próxima etapa aqueles documentos com status igual a "A";
- **Segunda etapa:** a etapa **\$group** agrupa os documentos pelo campo cust_id para calcular a soma do valor para cada cust_id exclusivo.

Framework de agregação utilizando Mongo Compass

The screenshot displays the MongoDB Compass web interface. At the top, a navigation bar includes tabs for Documents (25.4K), Aggregations (highlighted with a red box), Schema, Indexes (1), and Validation. Below this, a message states 'Your pipeline is currently empty. Need help getting started? [Generate aggregation](#)'. Action buttons for Explain, Export, Run, and Options are present. The main area shows 'Untitled' with buttons for SAVE, CREATE NEW, and EXPORT TO LANGUAGE. A 'PREVIEW' toggle is active. Below the document count '25359 Documents in the collection', a 'Preview of documents' section shows three document snippets. At the bottom, a '+ Add Stage' button is highlighted with a red box, with a link 'Learn more about aggregation pipeline stages' below it.

Documents 25.4K **Aggregations** Schema Indexes 1 Validation

Your pipeline is currently empty. Need help getting started? [Generate aggregation](#)

Explain Export Run Options

Untitled SAVE CREATE NEW EXPORT TO LANGUAGE PREVIEW STAGES TEXT WIZARD

25359 Documents in the collection

Preview of documents

```
{ "_id": ObjectId('5eb3d668b31de5d588f4292a'),  
  "address": Object,  
    "borough": "Brooklyn",  
    "cuisine": "American",  
  "grades": Array (4),  
  "name": "Riviera Caterer",  
  "restaurant_id": "40356018"  
}
```

```
{ "_id": ObjectId('5eb3d668b31de5d588f4292b'),  
  "address": Object,  
    "borough": "Brooklyn",  
    "cuisine": "Delicatessen",  
  "grades": Array (6),  
  "name": "Wilken'S Fine Food",  
  "restaurant_id": "40356483"  
}
```

```
{ "_id": ObjectId('5eb3d668b31de5d588f4292c'),  
  "address": Object,  
    "borough": "Staten Island",  
    "cuisine": "Jewish/Kosher",  
  "grades": Array (4),  
  "name": "Kosher Island",  
  "restaurant_id": "40356442"  
}
```

+ Add Stage

[Learn more about aggregation pipeline stages](#)

Steps

Preview

Framework de agregação

```
db.solarSystem.aggregate([{\n  "$match": {\n    "atmosphericComposition": { "$in": [/O2/] },\n    "meanTemperature": { $gte: -40, "$lte": 40 }\n  }\n}, {\n  "$project": {\n    "_id": 0,\n    "name": 1,\n    "hasMoons": { "$gt": ["$numberOfMoons", 0] }\n  }\n}]);
```

Output

[{ name: 'Earth', hasMoons: true }]

- Neste exemplo, `$match` e `$project` são operadores de agregação e `$in`, `$gte` e `$lte` são operadores de consulta;
- Na etapa de projeção (`$project`), `$gt` é uma expressão. Os seus argumentos são fornecidos no array.

Framework de agregação

- O operador `$match` filtra os documentos, enviando para a próxima etapa os documentos que correspondem às condições especificadas;
- Podemos utilizar o operador `$match` várias vezes;
- Podemos realizar correspondências com base em `comparação`, `lógica`, `arrays` e muito mais;

Framework de agregação - Match

- Se o operador `$match` for incluído na `primeira` etapa, pode tirar partido de `índices`, o que aumenta a `velocidade` das consultas;
- O `$match` deve ser incluído no `início` dos pipelines;
- O operador `$match` não possui nenhum mecanismo de `projeção`.

Framework de agregação - Match

- Exemplo:

```
// filtra todos os corpos celestiais que não são "Star"
db.solarSystem.aggregate([
  "$match": { "type": { "$ne": "Star" } }
]).pretty()
```

Framework de agregação - Match

- Exemplos:

```
// Contar o número de documentos (sem utilizar framework de agregação)  
db.solarSystem.count() ;
```

```
// utilizando o operador $count (utilizando framework de agregação)  
db.solarSystem.aggregate([{  
  "$match": { "type": { "$ne": "Star" } }  
}, {  
  "$count": "planets"  
}]);
```

Framework de agregação - Project

- Passa os documentos com os **campos solicitados** para a **próxima etapa** do pipeline;
- Os campos especificados podem ser campos **existentes** dos documentos de entrada ou campos **recém-calculados**.

Framework de agregação - Project

- Exemplos:

```
//Remove o campo: _id do documento projetado utilizando a função find
db.solarSystem.find({}, {"_id": 0, "name": 1});
```

```
// projeta o campo ``name`` e remove o campo ``_id`` com função aggregate
db.solarSystem.aggregate([{"$project": {"_id": 0, "name": 1 } }]);
```

```
// utilização da notação "dot" para expressar os campos para projeção
db.solarSystem.aggregate([
    {"$project": {"_id": 0, "name": 1,
                  "gravity.value": 1 }
    }
]);
```

Framework de agregação - Project

- Exemplos:

```
// Atribuir o valor da gravidade a um campo: "gravity"
```

```
db.solarSystem.aggregate([  
    {"$project": {  
        "_id": 0, "name": 1,  
        "gravity": "$gravity.value" }}]);
```

```
// Criar um novo campo: "myWeight" utilizando expressões
```

```
db.solarSystem.aggregate([  
    {"$project": { "_id": 0, "name": 1,  
        "myWeight": { "$multiply":  
            [ { "$divide":  
                [ "$gravity.value", 9.8 ] }, 86 ]  
            } }  
    }  
]);
```


Framework de agregação - Project

- Com este operador pode-se executar operações de **acumulação** (\$sum, \$avg, \$max, \$min) tal como o \$group.
- No entanto, enquanto o **\$group** opera sobre conjuntos de documentos, o \$project opera sobre um array (de um documento individual).
- Exemplo:

```
db.restaurants.aggregate([  
  {$project: {scoreMedio: {$avg: "$grades.score"}}}  
])
```

```
{ _id: ObjectId("61b1e344d55170cefd6c734a"), scoreMedio: 8.2 }  
{ _id: ObjectId("61b1e344d55170cefd6c734b"), scoreMedio: 13.75 }  
{ _id: ObjectId("61b1e344d55170cefd6c734c"), scoreMedio: 9.25 }
```

Framework de agregação - AddFields

- O operador de estágio `$addFields` é similar ao `$project`, mas mantém todos os campos do documento original. Este operador permite adicionar novos campos ou substituir campos existentes.

```
db.restaurants.aggregate([
  {
    $addFields: {
      scoreMedio:
        {$avg: "$grades.score"}
    }
  }
])
```

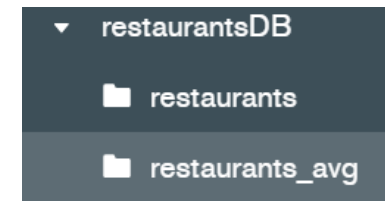


```
{ _id: ObjectId("61b1e344d55170cefd6c734a"),
  address:
    { building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462' },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades:
    [ { date: 2014-03-03T00:00:00.000Z, grade: 'A', score: 2 },
      { date: 2013-09-11T00:00:00.000Z, grade: 'A', score: 6 },
      { date: 2013-01-24T00:00:00.000Z, grade: 'A', score: 10 },
      { date: 2011-11-23T00:00:00.000Z, grade: 'A', score: 9 },
      { date: 2011-03-10T00:00:00.000Z, grade: 'B', score: 14 } ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445',
  scoreMedio: 8.2 }
```

Framework de agregação - Out

- O operador de estágio `$out` guarda os documentos resultantes da `pipeline` de agregação numa `coleção` especificada. Se essa coleção `já existe`, é `substituída` pelo resultado da pipeline. Caso contrário, cria uma nova coleção.
- O operador `$out` deverá corresponder ao último estágio da pipeline.
- Exemplo:

```
db.restaurants.aggregate([  
  {  
    $addFields: {scoreMedio:  
      {$avg: "$grades.score"}},  
    {$out: "restaurants_avg"}  
  ])
```



Framework de agregação – Sort, Skip e count

- `$sort`, `$skip`, `$limits` e `$count` são funcionalmente equivalentes aos métodos de cursor;
- O `$sort` necessita de um campo para critério de ordenação;
- Um aspeto importante para se referir é que a ordenação **não** se **limita** apenas a um **único** campo;
- Por padrão, a ordenação está limitada a **100 megabytes de RAM**. Se for necessária mais memória, é necessário fornecer a opção `allowDiskUse = true` para o pipeline.

Framework de agregação – Sort, Skip e count

- Exemplos - Sort:

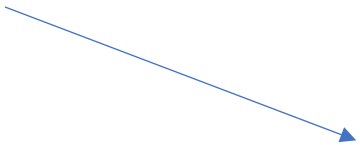
```
// ordena documentos com método de cursor
db.solarSystem.find({}, { "_id": 0,
                          "name": 1,
                          "numberOfMoons": 1 })
                          .sort( { "numberOfMoons": -1 } ).pretty();

// ``$sort`` stage
db.solarSystem.aggregate([ {
  "$project": {
    "_id": 0,
    "name": 1,
    "numberOfMoons": 1
  }
}, {
  "$sort": { "numberOfMoons": -1 }
}]).pretty();
```

Framework de agregação – Sort, Skip e count

- Exemplos - Sort:

```
// opção ``allowDiskUse``  
db.solarSystem.aggregate([  
  "$project": {  
    "_id": 0,  
    "name": 1,  
    "hasMagneticField": 1,  
    "numberOfMoons": 1  
  }  
], {  
  "$sort": { "hasMagneticField": -1, "numberOfMoons": -1 }  
}],  
{ "allowDiskUse": true });
```



permite a escrita de ficheiros temporários em disco quando um stage excede o limite de 100 megabytes

Framework de agregação – Sort, Skip e count

- Exemplos - Skip:

```
// descartar documentos utilizando método de cursor
db.solarSystem.find({}, {"_id": 0, "name": 1, "numberOfMoons":
1}).skip(5).pretty();
```

```
// ``skip`` stage
db.solarSystem.aggregate([ {
  "$project": {
    "_id": 0,
    "name": 1,
    "numberOfMoons": 1
  }
}, {
  "$skip": 1
}]).pretty()
```

Framework de agregação – Sort, Skip e count

- Exemplos - limit:

```
// limitar número de documentos utilizando método de cursor
db.solarSystem.find({}, {"_id": 0, "name": 1, "numberOfMoons": 1}).limit(5).pretty();
```

```
// ``$limit`` stage
db.solarSystem.aggregate([
  {
    "$project": {
      "_id": 0,
      "name": 1,
      "numberOfMoons": 1
    }
  },
  { "$limit": 5 }
]).pretty();
```


Framework de agregação – Sort, Skip e count

- Exemplos - count:

```
// ``$count`` stage
db.solarSystem.aggregate([ {
  "$match": {
    "type": "Terrestrial planet"
  }
}, {
  "$project": {
    "_id": 0,
    "name": 1,
    "numberOfMoons": 1
  }
}, {
  "$count": "terrestrial planets"
}]) .pretty();
```

Framework de agregação – Sample

- `$sample` é muito útil ao trabalhar com grandes coleções em que queremos apenas uma quantidade `limitada` de documentos para operar;
- Pode ser útil para realizar uma análise inicial ou para fazer alguma `amostragem` no conjunto de resultados;
- Exemplo:

```
db.nycFacilities.aggregate([{"$sample": { "size": 200 }}]).pretty();
```

Framework de agregação – Group

- Documentos com valores idênticos são agrupados e cada valor único produz um documento de saída que nos mostra os valores ou valores nos quais agrupamos;
- Podemos especificar campos adicionais que desejamos calcular na fase de grupos;
- `_id` é onde especificamos como os documentos devem ser agrupados;
- O `$group` pode ser utilizado várias vezes num pipeline.

Framework de agregação – Group

- Exemplo:

```
// Agrupa por ano e conta por ano
db.movies.aggregate([
  {
    "$group": {
      "_id": "$year",
      "numFilmsThisYear": { "$sum": 1 }
    }
  }
])
```

Framework de agregação – Group

- Exemplo:

```
// agrupar e ordenar por ordem descendente baseado na contagem
db.movies.aggregate([
  {
    "$group": {
      "_id": "$year",
      "count": { "$sum": 1 }
    }
  },
  {
    "$sort": { "count": -1 }
  }
])
```

Framework de agregação – Group

- Exemplo:

```
// agrupar todos os documentos. Por convenção utilizamos o valor null
db.movies.aggregate([
  {
    "$group": {
      "_id": null,
      "count": { "$sum": 1 }
    }
  }
])
```

Framework de agregação – Unwind

- `$unwind` faz com que os arrays sejam **decompostos** (descompactados), criando um novo documento para cada entrada onde o valor do campo é agora **cada valor** do array;
- O operador `$unwind` só funciona num **array**;
- O **descompactar** em grandes coleções com grandes documentos pode levar a problemas de desempenho.

Framework de agregação – Unwind

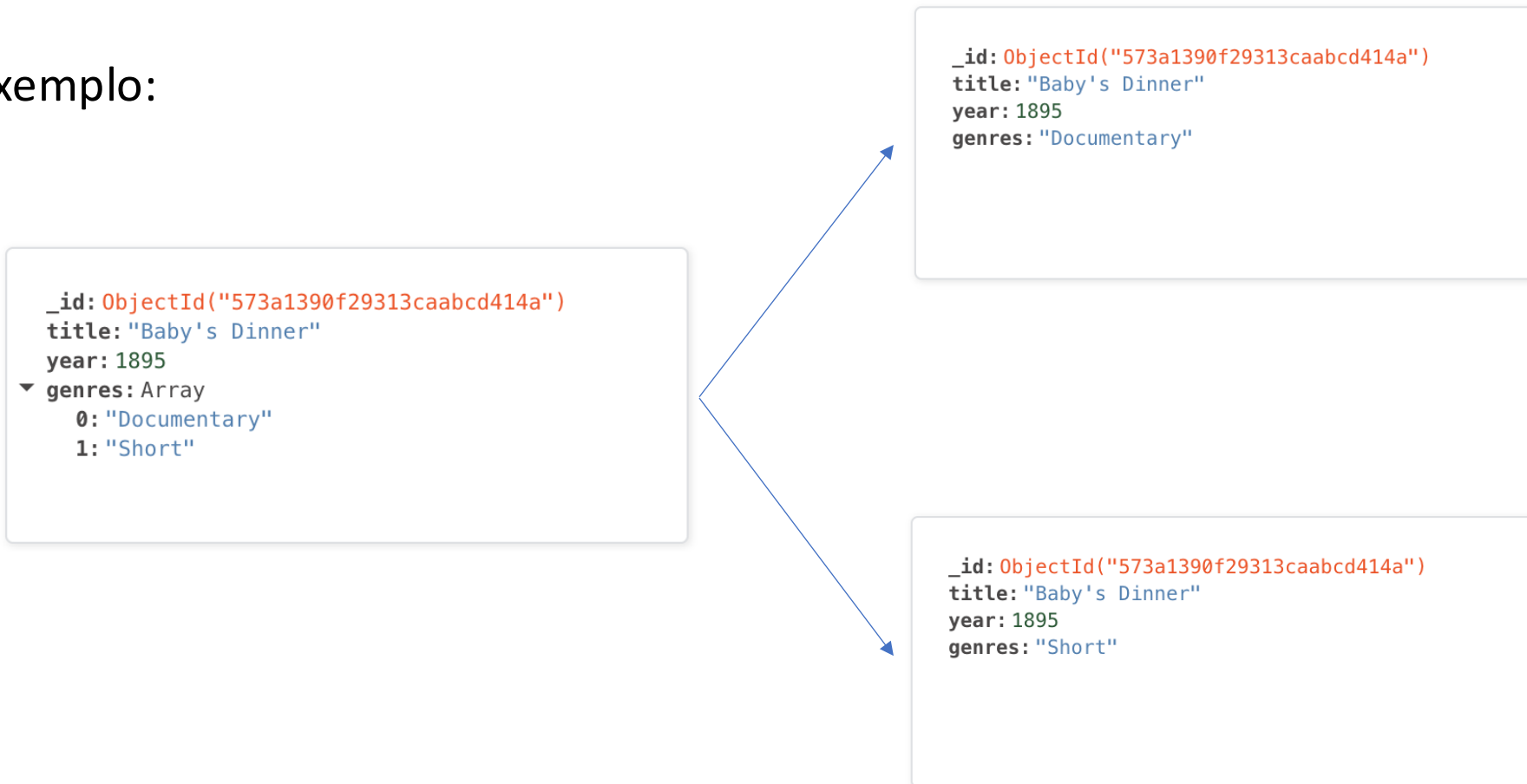
■ Exemplo:

```
_id: ObjectId("573a1390f29313caabcd414a")
title: "Baby's Dinner"
year: 1895
runtime: 1
released: 1895-12-28T00:00:00.000+00:00
> cast: Array
plot: "A baby is seated at a table between its cheerful parents, Auguste and ..."
fullplot: "A baby is seated at a table between its cheerful parents, Auguste and ..."
lastupdated: "2015-08-12 00:06:40.657000000"
type: "movie"
> directors: Array
> imdb: Object
> countries: Array
✓ genres: Array
  0: "Documentary"
  1: "Short"
```

Podemos realizar a
decomposição

Framework de agregação – Unwind

■ Exemplo:



Framework de agregação – Unwind

```
[{
  $match: {
    _id: ObjectId("573a1390f29313caabcd414a")
  }
}, {
  $project: {
    title: 1,
    year: 1,
    genres: 1
  }
}, {
  $unwind: {
    path: "$genres"
  }
}]
```

The diagram illustrates the MongoDB aggregation framework's `$unwind` stage. It shows a document with a single `genres` array being transformed into two separate documents, one for each genre.

The initial document (left) has the following structure:

```
{
  _id: ObjectId("573a1390f29313caabcd414a"),
  title: "Baby's Dinner",
  year: 1895,
  genres: ["Documentary", "Short"]
}
```

The `$unwind` stage (middle) is applied with the path `"$genres"`. This stage decomposes the array into individual documents. The resulting documents (right) are:

```
{
  _id: ObjectId("573a1390f29313caabcd414a"),
  title: "Baby's Dinner",
  year: 1895,
  genres: "Documentary"
}
```

```
{
  _id: ObjectId("573a1390f29313caabcd414a"),
  title: "Baby's Dinner",
  year: 1895,
  genres: "Short"
}
```

Framework de agregação – Unwind

// quantidade de género de filmes por ano entre 2010 e 2015

```
db.movies.aggregate([
  {
    "$match": {
      "year": { "$gte": 1995, "$lte": 2015 } }
  },
  {
    "$unwind": "$genres"
  },
  {
    "$group": {
      "_id": {
        "year": "$year",
        "genre": "$genres"
      },
      "average_rating": { "$avg": "$imdb.rating" }
    },
  },
  {
    "$sort": { "_id.year": -1, "average_rating": -1 }
  }
])
```

No documento original, os géneros são armazenados num array:

```
"genres": [
  "Documentary",
  "Short"
]
```

Framework de agregação – Lookup

- Este operador representa uma **junção** entre **duas coleções**;
- Para cada documento de **entrada**, a etapa **\$lookup** adiciona um novo campo do tipo array cujos elementos são os documentos correspondentes da coleção “unida”;

Framework de agregação – Lookup

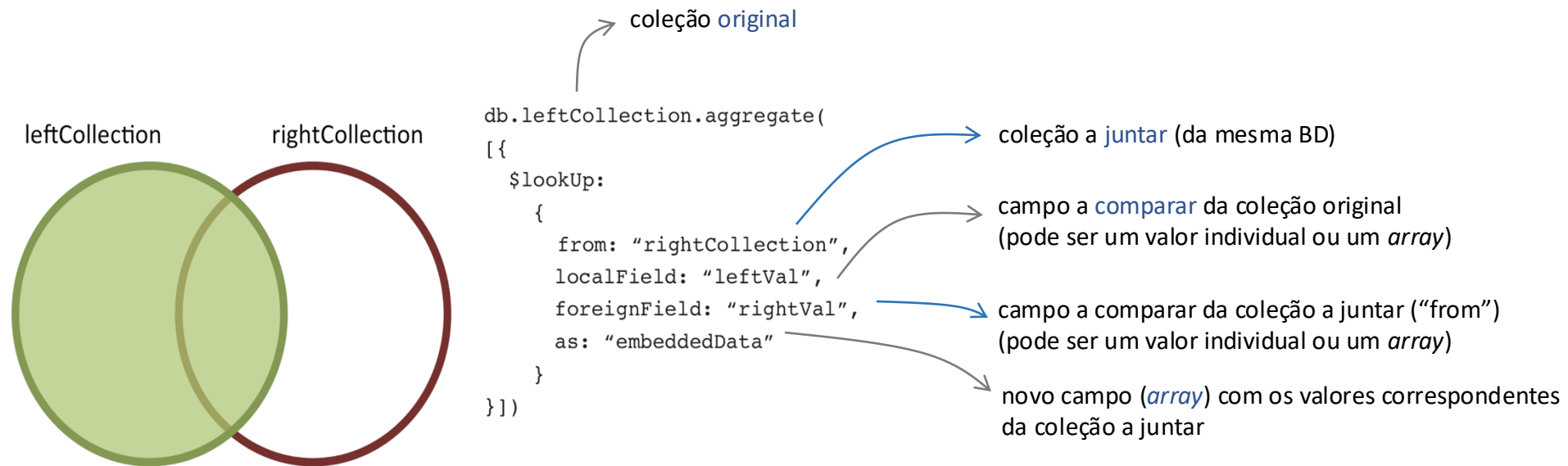


Imagem retirada de: <https://www.mongodb.com/blog/post/joins-and-other-aggregation-enhancements-coming-in-mongodb-3-2-part-1-of-3-introduction>

Framework de agregação – Lookup

```
// lookup que realiza a junção da coleção
air_alliances com a coleção air_airlines
db.air_alliances
  .aggregate([
    {
      "$lookup": {
        "from": "air_airlines",
        "localField": "airlines",
        "foreignField": "name",
        "as": "airlines"
      }
    }
  ])
.pretty()
```



```
_id: ObjectId("5980bef9a39d0ba3c650ae9b")
name: "Star Alliance"
▼ airlines: Array
  ▼ 0: Object
    _id: ObjectId("56e9b497732b6122f87903ca")
    airline: 330
    name: "Air Canada"
    alias: "AC"
    iata: "ACA"
    icao: "AIR CANADA"
    active: "Y"
    country: "Canada"
    base: "TAL"
  ► 1: Object
```

Framework de agregação – Lookup

- A etapa `$lookup` envia esses documentos remodelados para a próxima etapa.
- O campo `from` é a coleção da qual desejamos consultar os documentos.
- A coleção especificada no campo `from` não pode ser fragmentada e deve existir na mesma base de dados.

Framework de agregação – Lookup

- Todas as correspondências serão colocadas num **array**;
- Se não existirem correspondências, o campo conterá um **array vazio**;
- Os valores de **localField** e **ForeignField** são correspondidos através de igualdade;

Framework de agregação – Lookup

■ Exemplo:

```
_id: ObjectId("5980bef9a39d0ba3c650ae9b")
name: "Star Alliance"
airlines: Array
  0: "Air Canada"
  1: "Adria Airways"
  2: "Avianca"
  3: "Scandinavian Airlines"
  4: "All Nippon Airways"
  5: "Brussels Airlines"
  6: "Shenzhen Airlines"
  7: "Air China"
  8: "Air New Zealand"
  9: "Asiana Airlines"
  10: "Copa Airlines"
  11: "Croatia Airlines"
  12: "EgyptAir"
  13: "TAP Portugal"
  14: "United Airlines"
  15: "Turkish Airlines"
  16: "Swiss International Air Lines"
  17: "Lufthansa"
  18: "EVA Air"
  19: "South African Airways"
  20: "Singapore Airlines"
```

Para cada valor procura
uma correspondência
na coleção air_airlines

>

```
_id: ObjectId("56e9b497732b6122f8790280")
airline: 4
name: "2 Sqn No 1 Elementary Flying Training School"
alias: ""
iata: "WYT"
icao: ""
active: "N"
country: "United Kingdom"
base: "HGH"
```

```
_id: ObjectId("56e9b497732b6122f8790281")
airline: 2
name: "135 Airways"
alias: ""
iata: "GNL"
icao: "GENERAL"
active: "N"
country: "United States"
base: "LHE"
```

Transformação/migração de dados

- Utilizando as funções do mongo (de cursor e aggregate) podemos **migrar** dados entre coleções e realizar alterações à **estrutura** dos documentos da coleção;
- Desta forma, podemos **importar** dados com uma dada estrutura e transformá-los para uma estrutura mais adequada para as consultas que se pretende realizar.

Transformação/migração de dados

```
> _id: ObjectId("5ff323e03dbd8828f8c6f5a4")  
ReceiptID: "43659"  
OrderDate: "2011-05-31 00:00:00.000"  
Customer: "29825"  
CurrencyRateID: "NULL"  
SubTotal: "20565.6206"  
TaxAmt: "1971.5149"  
Store: "1046"  
StoreName: "Better Bike Shop"  
ReceiptLineID: "1"  
Quantity: "1"  
ProductID: "776"  
UnitPrice: "2024.994"  
LineTotal: "2024.994"
```

```
_id: ObjectId("5ff323e03dbd8828f8c6f5a5")  
ReceiptID: "43659"  
OrderDate: "2011-05-31 00:00:00.000"  
Customer: "29825"  
CurrencyRateID: "NULL"  
SubTotal: "20565.6206"  
TaxAmt: "1971.5149"  
Store: "1046"  
StoreName: "Better Bike Shop"  
ReceiptLineID: "2"  
Quantity: "3"  
ProductID: "777"  
UnitPrice: "2024.994"  
LineTotal: "6074.982"
```

São linhas de venda
que se referem à
mesma venda. Como
poderíamos agrupar
num só documento
com um array de
“linhas de fatura”?

Transformação/migração de dados

- Exemplo de consulta:

```
db.salesDetails.aggregate(  
[  
  {  
    $group: {  
      _id: { ReceiptID: "$ReceiptID" },  
      products: { $push: { item: "$ProductID" } }  
    }  
  }  
])
```

Acumulador:

<https://docs.mongodb.com/manual/reference/operator/aggregation/group/#accumulators-group>

Transformação/migração de dados

- Output:

```
[
  { _id: '43659',
    products: [
      { item: '776' },
      { item: '777' },
      { item: '778' },
      { item: '771' },
      { item: '772' },
      { item: '773' },
      { item: '774' },
      { item: '714' },
      { item: '716' },
      { item: '709' },
      { item: '712' },
      { item: '711' }
    ]
  }
]
```

Transformação/migração de dados

- Também é possível devolver o **iterator** de uma pipeline de agregação (embora tenha efeitos negativos no desempenho):

```
// Definir a pipeline de agregação
const pipeline = [
  { $match: { status: "active" } },
  { $group: { _id: "$category", total: { $sum: "$value" } } }
];

// Executar a agregação e guardar o cursor
const cursor = db.nome_da_colecao.aggregate(pipeline);

// Utilizar o cursor como iterator
cursor.forEach(doc => printjson(doc));
```

Bibliografia/Referências

- <https://www.mongodb.com/pt-br/docs/manual/aggregation/>
- Phaltankar, A. Ahsan, J., Harrison, M., Nedov, L. (2020). MongoDB Fundamentals: A hands-on guide to using MongoDB and Atlas in the real world. Packt Publishing

MongoDB: Framework de Agregação