

Agregação &gt; Pipeline de agregação

# Pipeline de agregação

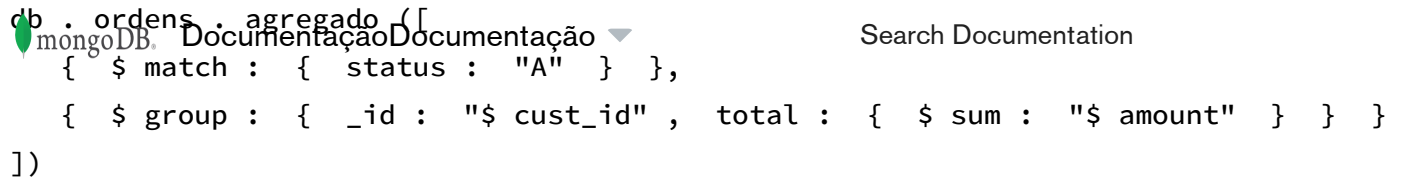
## Nesta página

- Pipeline
- Expressões de pipeline
- Comportamento do pipeline de agregação
- Considerações

O pipeline de agregação é uma estrutura para agregação de dados modelada no conceito de pipelines de processamento de dados. Os documentos entram em um pipeline de várias etapas que transforma os documentos em resultados agregados. Por exemplo:

0:00 / 0:12

No exemplo,



```

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

```

**Primeira etapa** : a `$match` etapa filtra os documentos pelo `status` campo e passa para a próxima etapa os documentos `status` iguais "A".

**Segunda etapa** : a `$group` etapa agrupa os documentos por `cust_id` campo para calcular a soma do valor de cada único `cust_id`.

## Pipeline

The MongoDB aggregation pipeline consists of stages. Each stage transforms the documents as they pass through the pipeline. Pipeline stages do not need to produce one output document for every input document; e.g., some stages may generate new documents or filter out documents.

Pipeline stages can appear multiple times in the pipeline with the exception of `$out`, `$merge`, and `$geoNear` stages. For a list of all available stages, see [Aggregation Pipeline Stages](#).

MongoDB provides the `db.collection.aggregate()` method in the mongo shell and the `aggregate` command to run the aggregation pipeline.

For example usage of the aggregation pipeline, consider [Aggregation with User Preference Data](#) and [Aggregation with the Zip Code Data Set](#).

Starting in MongoDB 4.2, you can use the aggregation pipeline for updates in:

Command	mongo Shell Methods
<code>findAndModify</code>	<code>db.collection.findAndModify()</code>
	<code>db.collection.findOneAndUpdate()</code>

update

`db.collection.updateOne()``db.collection.updateMany()``db.collection.update()`

## Pipeline Expressions

Some pipeline stages take a pipeline expression as the operand. Pipeline expressions specify the transformation to apply to the input documents. Expressions have a document structure and can contain other expression.

Pipeline expressions can only operate on the current document in the pipeline and cannot refer to data from other documents: expression operations provide in-memory transformation of documents.

Generally, expressions are stateless and are only evaluated when seen by the aggregation process with one exception: accumulator expressions.

The accumulators, used in the `$group` stage, maintain their state (e.g. totals, maximums, minimums, and related data) as documents progress through the pipeline.

*Changed in version 3.2:* Some accumulators are available in the `$project` stage; however, when used in the `$project` stage, the accumulators do not maintain their state across documents.

For more information on expressions, see [Expressions](#).

## Aggregation Pipeline Behavior

In MongoDB, the `aggregate` command operates on a single collection, logically passing the *entire* collection into the aggregation pipeline. To optimize the operation, wherever possible, use the following strategies to avoid scanning the entire collection.

### Pipeline Operators and Indexes

The `$match` and `$sort` pipeline operators can take advantage of an index when they occur at the **beginning** of the pipeline.

The `$geoNear` pipeline operator takes advantage of a `geo` index. The `$geoNear` pipeline operation must appear as the first stage in the aggregation pipeline.

[Search Documentation](#)

*Changed in version 3.2:* Starting in MongoDB 3.2, indexes can cover an aggregation pipeline. In MongoDB 2.6 and 3.0, indexes could not cover an aggregation pipeline since even when the pipeline uses an index, aggregation still requires access to the actual documents.

## Early Filtering

If your aggregation operation requires only a subset of the data in a collection, use the `$match`, `$limit`, and `$skip` stages to restrict the documents that enter at the beginning of the pipeline. When placed at the beginning of a pipeline, `$match` operations use suitable indexes to scan only the matching documents in a collection.

Placing a `$match` pipeline stage followed by a `$sort` stage at the start of the pipeline is logically equivalent to a single query with a sort and can use an index. When possible, place `$match` operators at the beginning of the pipeline.

## Considerations

### Sharded Collections

The aggregation pipeline supports operations on sharded collections. See [Aggregation Pipeline and Sharded Collections](#).

### Aggregation vs Map-Reduce

The aggregation pipeline provides an alternative to map-reduce and may be the preferred solution for aggregation tasks where the complexity of map-reduce may be unwarranted.

### Limitations

O pipeline de agregação tem algumas limitações nos tipos de valor e tamanho do resultado. Consulte [Limites do pipeline de agregação](#) para obter detalhes sobre limites e restrições no pipeline de agregação.

### Otimização de pipeline

O pipeline de agregação possui uma fase de otimização interna para determinadas sequências de operadores. Para detalhes sobre a otimização de agregação, consulte a documentação.

[Search Documentation](#)