Aggregation

On this page

Aggregation framework operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result.

The Aggregation Pipeline

The aggregation pipeline is a framework for data aggregation modeled on the concept of data processing pipelines. Documents enter a multi-stage pipeline that transforms the documents into aggregated results.

For a full explanation and a complete list of pipeline stages and operators, see the manual.

The following example uses the aggregation pipeline on the `restaurants` sample dataset to find a list of the total number of 5-star restaurants, grouped by restaurant category.

```
client = Mongo::Client.new([ '127.0.0.1:27017' ], :database => 'test')
coll = client['restaurants']
aggregation = coll.aggregate([
       { '$match'=> { 'stars'=> 5 } },
       { '$unwind'=> '$categories'},
       { '$group'=> { '_id'=> '$categories', 'fiveStars'=> { '$sum'=> 1 } } }
    ])

aggregation.each do |doc|
  #=> Yields a BSON::Document.
end
```

Inside the `aggregate` method, the first pipeline stage filters out all documents except those with `5` in the `stars` field. The second stage unwinds the `categories` field, which is an array, and treats each item in the array as a separate document. The third stage groups the documents by category and adds up the number of matching 5-star results.

Aggregation pipeline stages have a maximum memory use limit. To handle large datasets, set the `allowDiskUse` option to true to enable writing data to temporary files.

- You can call the `allow_disk_use` method the `aggregation` object to get a new object with the option set:

```
aggregation = coll.aggregate([ <aggregration pipeline expressions> ])
aggregation_with_disk_use = aggregation.allow_disk_use(true)
```

- Or you can pass an option to the `aggregate` method:

```
aggregation = coll.aggregate([ <aggregration pipeline expressions> ],
                              :allow_disk_use => true)
```

Single Purpose Aggregation Operations

MongoDB provides helper methods for some aggregation functions, including count and distinct.

Count

The following example demonstrates how to use the `count` method to find the total number of documents which have the exact array `[ 'Chinese', 'Seafood' ]` in the `categories` field.

```
client = Mongo::Client.new([ '127.0.0.1:27017' ], :database => 'test')
coll = client['restaurants']
aggregation = coll.count({ 'categories': [ 'Chinese', 'Seafood' ] })

count = coll.count({ 'categories' => [ 'Chinese', 'Seafood' ] })
```

Distinct

The `distinct` helper method eliminates results which contain values and returns one record for each unique value.

The following example returns a list of unique values for the `categories` field in the `restaurants` collection:

```
client = Mongo::Client.new([ '127.0.0.1:27017' ], :database => 'test')
coll = client['restaurants']
aggregation = coll.distinct('categories')

aggregation.each do |doc|
  #=> Yields a BSON::Document.
end
```