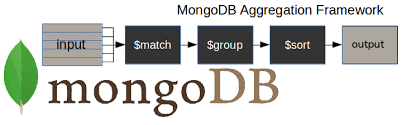
**Aggregation at MongoDB**

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

MongoDB provides aggregation as :

1. [aggregation pipeline](https://docs.mongodb.com/manual/aggregation/#aggregation-framework)
2. [map-reduce function](https://docs.mongodb.com/manual/aggregation/#aggregation-map-reduce)
3. [single purpose aggregation methods](https://docs.mongodb.com/manual/aggregation/#single-purpose-agg-operations).

[](https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjnnb3NleXYAhVJaRQKHVt4CXoQjRwIBw&url=http%3A%2F%2Fjuanroy.es%2F2015%2F06%2F&psig=AOvVaw1SRaFkXD0xvSo4xL_vTqK5&ust=1516490412863084)

**Single Puprose Aggregation Methods**

these operations aggregate documents from a single collection. While these operations provide simple access to common aggregation processes

db.orders.count()

db.orders.count({status:"d"})

db.orders.distinct("custid")

db.orders.distinct("custid").slice(0,2) // starts from 0 th index and upto 2-1 index

[**Aggregation pipeline**](https://docs.mongodb.com/manual/aggregation/#aggregation-framework)

Uses pipelines to process the Data

Documents enter a multi-stage pipeline that transforms the documents into an aggregated result.

The basic pipeline stages provide filters, document transformations

Other pipeline operations provide tools for grouping and sorting documents by specific field or fields as well as tools for aggregating the contents of arrays, including arrays of documents.

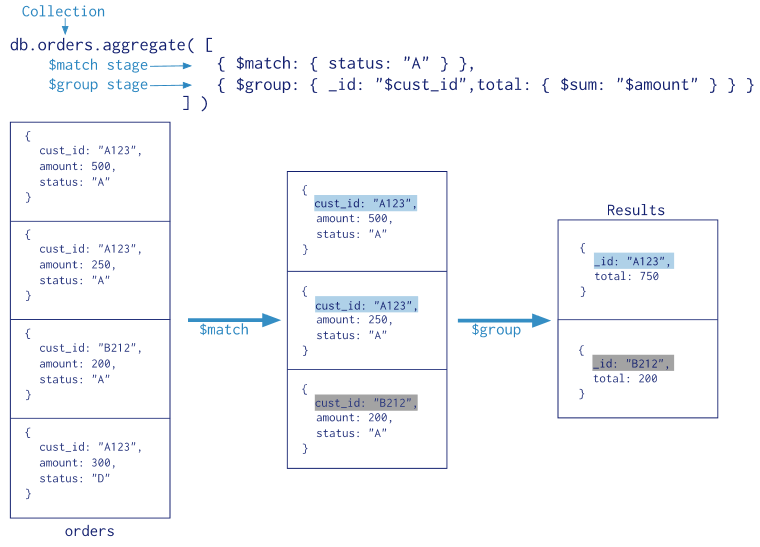
The aggregation pipeline can work on Sharded collection also

The aggregation pipeline can use indexes to improve its performance during some of its stages.

Syntax : db.CollectionName.aggregate([{firstAggregation}, {SecondAggregation},… ], {aggOptions})

We can see All the aggregation operators

<https://docs.mongodb.com/manual/reference/operator/aggregation/>



Example

db.Orders.find().count()

db.Orders.find({},{\_id:0,custid:0}).sort({"amount":-1})

db.Orders.aggregate([{$group:{\_id:"Sum",total:{$sum:"$amount"}}}])

db.Orders.aggregate([{$group:{\_id:"Sum",total:{$sum:"$amount"},count:{$sum:6}}}])

db.Orders.aggregate([{$group:{\_id:"$custid", total:{$sum:"$amount"}}}])

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

db.Orders.aggregate([{$group: {\_id:"$custid", total : {$sum: "$amount"}}},{$match: {total :{$gt:250}}}])

**Limitations**

Each Pipeline results in generating a cursor or document . There can more than one document for a pipeline , But the resulted document will be a BSON document , whose size limit is 16MB.

Pipeline stages have a limit of 100 megabytes of RAM. If a stage exceeds this limit, MongoDB will produce an error. To allow for the handling of large datasets, use the allowDiskUse option to enable aggregation

To allow Disk Usage:

db.stocks.aggregate( [

{ $project : { cusip: 1, date: 1, price: 1, \_id: 0 } },

{ $sort : { cusip : 1, date: 1 } }

],

{ allowDiskUse: true }

)

**Aggregation using Map -Reduce :**

map-reduce operations have two phases:

a map stage that processes each document and emits one or more objects for each input document,

and reduce phase that combines the output of the map operation.

Map-reduce uses custom JavaScript functions to perform the map and reduce operations

While the custom JavaScript provide great flexibility compared to the aggregation pipeline, in general, map-reduce is less efficient and more complex than the aggregation pipeline.

db.Orders.mapReduce(

function(){

emit(this.custid, this.amount)

},

function(key, values){

return Array.sum(values)

}

,

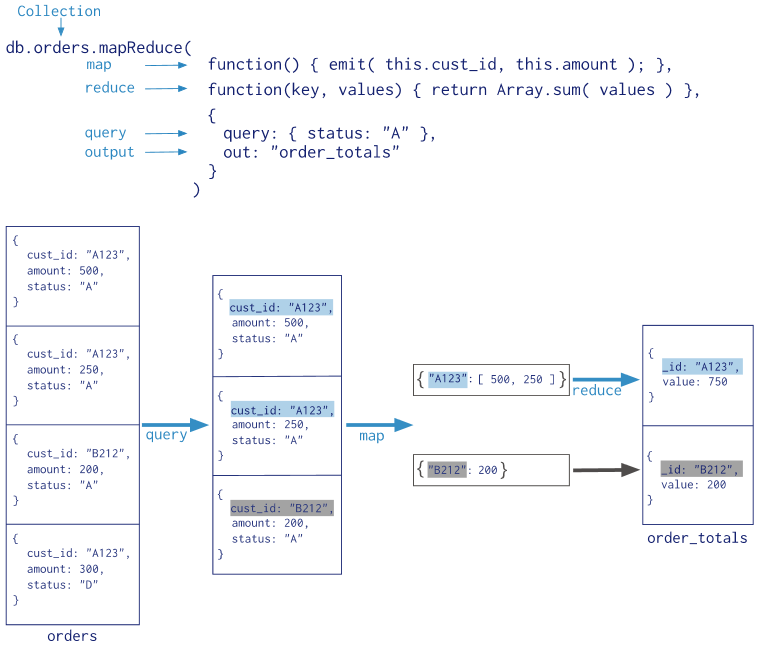
{

query:{status:"A"},

out:"amount"

}

)



When Map-reduce is used it work with JSON data , So the documents which are in the BSON Format Need to be converted into JSON Format and the aggregation need to be takes place .. So when compared to Aggregation Pipeline which uses the BSON directly , the map-Reduce performs slow