


# Aggregating Data

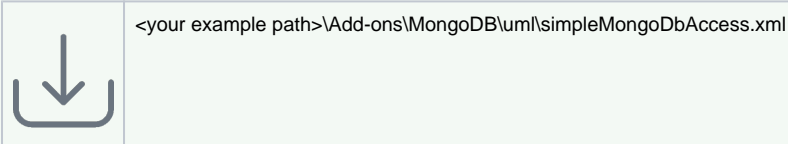
 This page explains the **MongoDB Adapter** in Bridge context. If you were looking for the same information regarding the **PAS Designer**, refer to **MongoDB Adapter** in the Designer guide.

Using a MongoDB aggregation pipeline, you can select and aggregate documents. A pipeline is an array of one or multiple stages that will be processed one after the other. Refer to the [MongoDB Manual](#) for more information on aggregation and pipelines.

The example below shows a simple aggregation pipeline that consists of two stages:

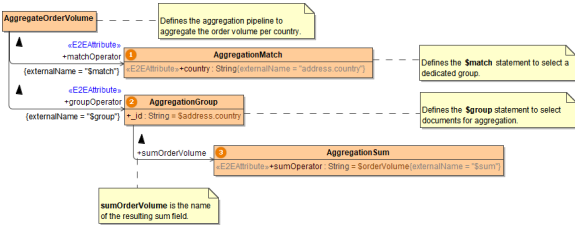
- a `$match` stage that selects documents from the database
- a `$group` stage that groups documents and summarizes a value

### Example File (Builder project Add-ons/MongoDB):



## Creating an Aggregation Pipeline

Aggregation stages can be reflected in MagicDraw using the following class construct:



The displayed class diagram defines aggregations stages to aggregate property **orderVolume** per **country** for all or a selected country.

Class	Description
1	<b>Stage \$match</b> Describes a match stage. <ul style="list-style-type: none"><li>• As a class property cannot have a name <code>\$match</code>, you need to apply stereotype <code>&lt;&lt;E2EAttribute&gt;&gt;</code> and external names <b>\$match</b>.</li><li>• The attribute of type <b>AggregationMatch</b> gives the name and value of the document property to select by. In this case, this is <b>address.country</b>, (attribute <b>country</b> with the corresponding external name) and its value.</li></ul>
2	<b>Stage \$group</b> Describes a group stage. <ul style="list-style-type: none"><li>• As a class property cannot have a name <code>\$group</code>, you need to apply stereotype <code>&lt;&lt;E2EAttribute&gt;&gt;</code> and external names <b>\$group</b>.</li><li>• Attribute <b>_id</b> is fix and contains the name of the property to group by. In this example, this is <b>\$address.country</b> which is set as a default value to the <b>_id</b> attribute.</li></ul>
3	<b>Sum Operator</b> <ul style="list-style-type: none"><li>• The structure below the <b>\$group</b> key defines the <code>\$sum</code> part of the grouping.</li><li>• The sum operator (<b>sumOperator</b> with external name <b>\$sum</b>) contains the name of the document property to summarize. In this example the property to summarize is fix, so it is given as a default value (<b>\$orderVolume</b>).</li></ul>

### On this Page:

- [Creating an Aggregation Pipeline](#)
- [Aggregating Data](#)

### Related Pages:

- [MongoDB Components](#)
- [Querying MongoDB](#)
- [Updating MongoDB Documents](#)
- [Aggregating Data](#)
- [Inserting and Deleting Documents](#)
- [MongoDB Adapter Reference](#)

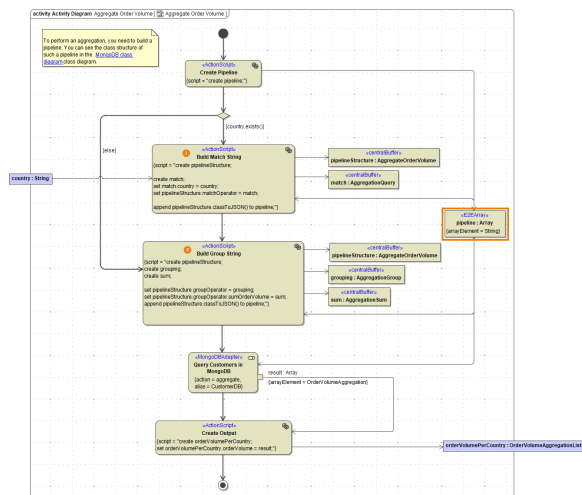
### Related Documentation:

- [MongoDB Manual](#)
  - [Aggregation](#)

You can add other stages to this diagram using the same pattern.

## Aggregating Data

The activity diagram below shows how to build the pipeline and trigger the aggregation.



Step	Description
1	<b>Build the Match Stage</b> <ul style="list-style-type: none"> <li>Create an object of the pipeline structure you have defined before. In this example, this is <b>pipelineStructure : AggregateOrderVolume</b>.</li> <li>Create the <b>\$match</b> stage.</li> <li>Set the value to compare with to the match expression.</li> <li>Append the stage to the <b>pipeline</b> array.</li> </ul>
2	<b>Build the Group Stage</b> <ul style="list-style-type: none"> <li>Create an object of the pipeline structure you have defined before. In this example, this is <b>pipelineStructure : AggregateOrderVolume</b>.</li> <li>Create the <b>\$group</b> stage and it's contained <b>\$sum</b> operator.</li> <li>Append the stage to the <b>pipeline</b> array.</li> </ul>

As a result, you will get a JSON document **orderVolumePerCountry** that contains the following order value aggregation:

```

{
  "orderVolume": [
    {
      "_id": "USA",
      "sumOrderVolume": 1098.0
    },
    {
      "_id": "CAN",
      "sumOrderVolume": 180.0
    }
  ]
}

```