CSCI235 Database Systems

MongoDB Aggregation Framework

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

MongoDB Aggregation Framework

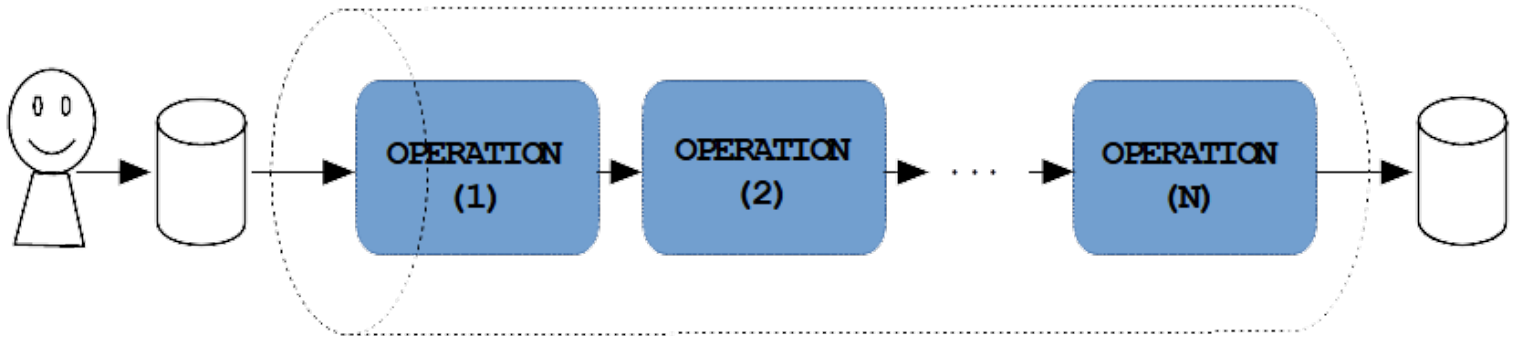
Outline

Aggregation framework ? What is it ?

Aggregation framework is a query language that that can be used to transform and to combine data from multiple documents in order to generate new information not available in any single document

Aggregation framework makes a task of database search much easier and more efficient through specification of a series of operations in an array and processing it in a single call

Aggregation framework defines an aggregation pipeline where the output from each step in the pipeline provides input to the next step



Aggregation Pipeline

Aggregation framework ? What is it ?

Every step in a pipeline executes a single operation on the input documents to transform the input and to generate output document

A pipeline processes a stream of documents through several operations like filtering, projecting, grouping, sorting, limiting, skipping, and the others

The same operations can be repeated many times in a pipeline in any order

Aggregation framework in MongoDB is similar to SQL `WITH` clause of `SELECT` statement

Some of the aggregation operators that can be used in an aggregation pipeline in MongoDB are similar to SQL `SELECT`, `WHERE`, `GROUP BY`, `HAVING`, `ORDER BY`, and `JOIN` clauses

Pipelined data processing is on of two basic way how data processing can be parallelised

The other way is partitioned data processing

Aggregation framework

Some of the operators that can be used in an aggregation pipeline:

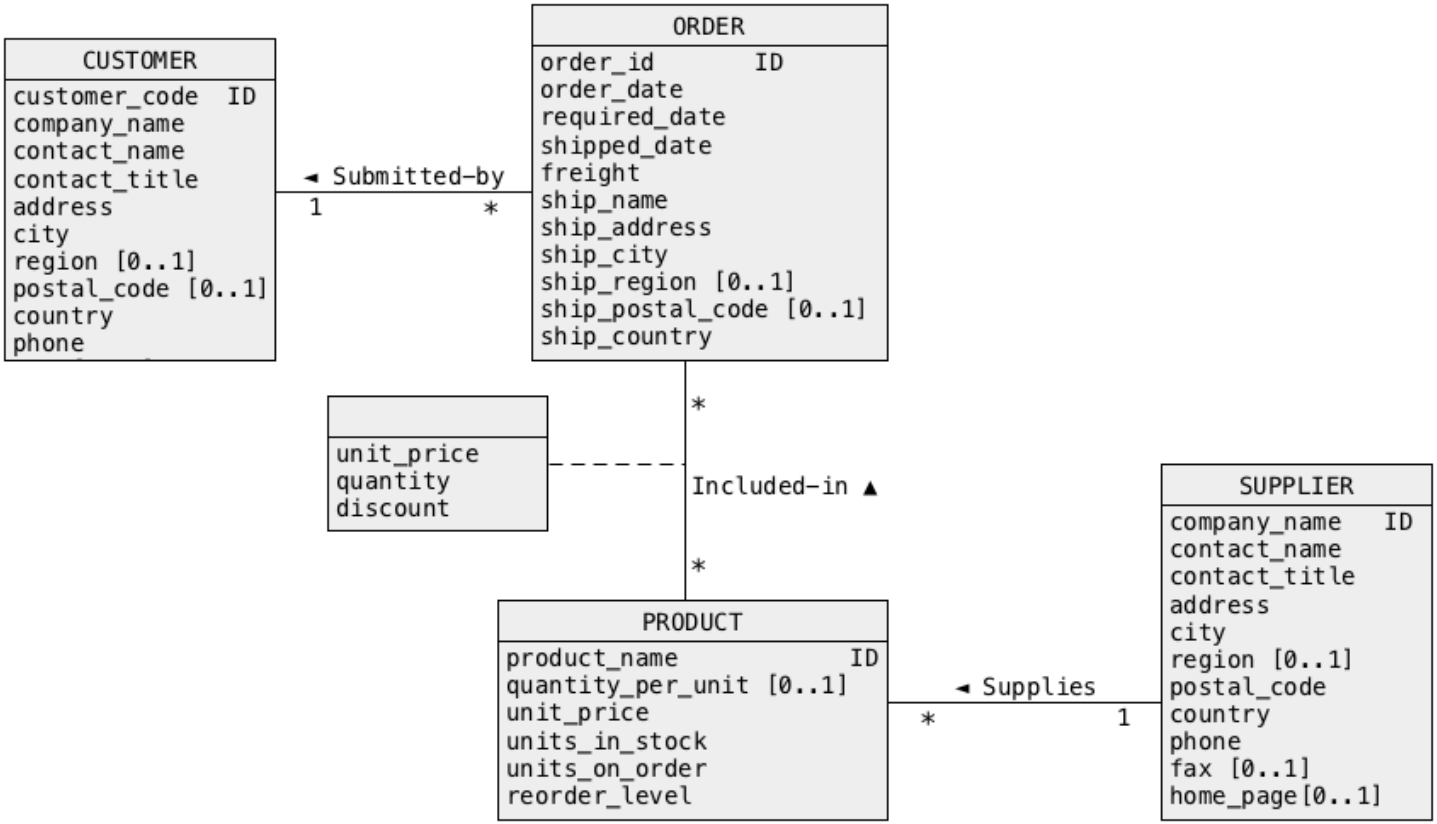
- `$project`: Extracts the components of a documents to be placed in an output document (similar to `SELECT` clause)
- `$match`: Filters the documents to be processed, similar to `find()` (and similar to `WHERE` clause)
- `$limit` and `$skip`: Limits and skips the documents to be passed to the next operation
- `$unwind`: Expands (unnest) an array, generating one output document for each array entry
- `$group`: Groups documents by a specified key
- `$sort` and `$count`: Sorts and counts the documents
- `$out`: Saves the results from a pipeline to a collection
- `$lookup`: Joins two collections of documents
- `$merge`: Merges two collections of documents

MongoDB Aggregation Framework

Outline

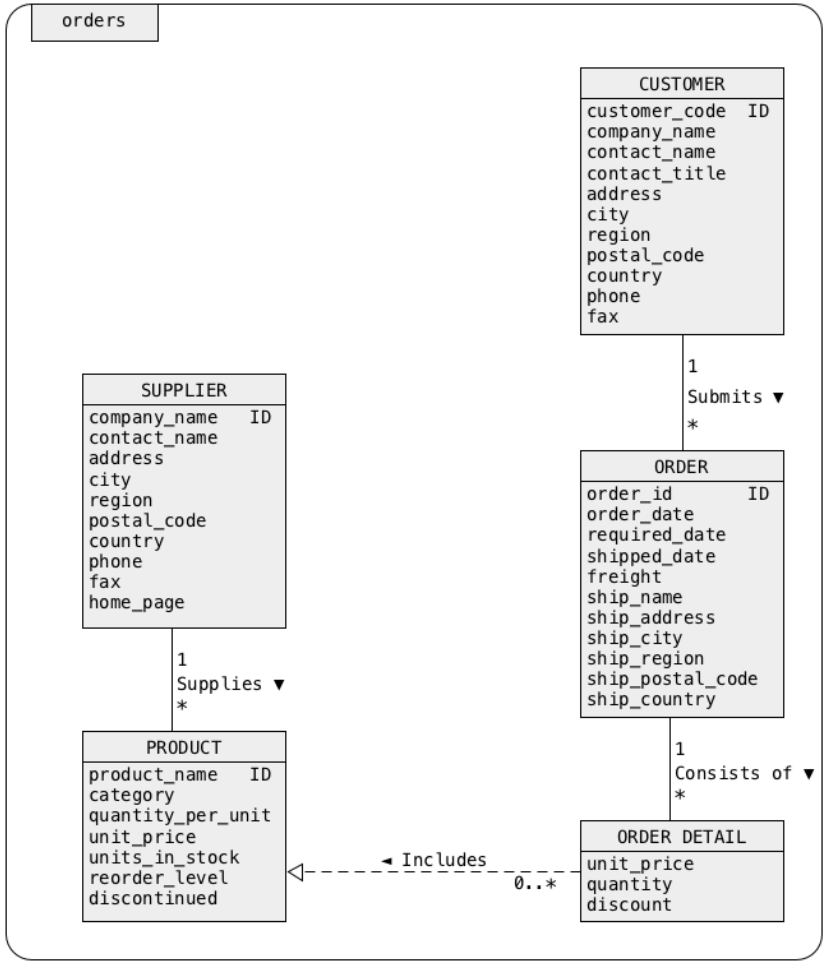
A sample database

A conceptual schema of a database with information about suppliers, products, customers, orders, and details of orders



A sample database

A sample collection `orders`



A sample database

A sample document, that belongs to a class `CUSTOMER`

CUSTOMER

```
{
        "_id":"ALFKI",
        "CUSTOMER":{
                "customer code":"ALFKI",
                "company name":"Alfreds Futterkiste",
                "contact name":"Maria Anders",
                "contact title":"Sales Representative",
                "address":"Obere Str. 57",
                "city":"Berlin",
                "region":null,
                "postal code":"12209",
                "country":"Germany",
                "phone":"030-0074321",
                "fax":"030-0076545",
                "submits":[ ]
        }
}
```

A sample database

A sample nested document, that belongs to a class `CUSTOMER`

CUSTOMER

```
{
 "_id":"FAMIA",
 "CUSTOMER":{
              "customer code":"FAMIA",
               ... ... ... ... ... ...
              "submits":[
                          {
                           "ORDER":{
                                     "order id":328,
                                      ... ... ... ... ...
                                     "consists of":[
                                                     {
                                                      "ORDER DETAIL":{
                                                                      "product name":"Louisiana Fiery Hot Pepper Sauce",
                                                                      ... ... ... ... ...
                                                                     }
                                                     },
                                                     {
                                                      "ORDER DETAIL":{
                                                                      "product name":"Raclette Courdavault",
                                                                      ... ... ... ... ...
                                                                     }
                                                     }
                                                    ]
                                    }
                           }
                          ]
             }
}
```

A sample database

A sample nested document, that belongs to a class `SUPPLIER`

SUPPLIER

```
{
 "_id":"Karkki Oy",
 "SUPPLIER":{
              "company name":"Karkki Oy",
              "contact name":"Anne Heikkonen",
              "contact title":"Product Manager",
              "address":"Valtakatu 12",
              ... ... ... ... ... ...
              "supplies":[
                           {
                            "PRODUCT":{
                                        "product name":"Maxilaku",
                                        "category name":"Confections",
                                         ... ... ... ... ...
                                       }
                           },
                           {
                            "PRODUCT":{
                                        "product name":"Valkoinen suklaa",
                                        "category name":"Confections",
                                        ... ... ... ... ...
                                       }
                           }

                          ]
             }
}
```

MongoDB Aggregation Framework

Outline

$project

Operator `$project` extracts components of subdocuments, renames components, and performs operations on components

Select company name of each customer and skip document identifier

aggregate($project)

```
db.orders.aggregate([ {$project:{"CUSTOMER.company name":1,"_id":0}} ])
```

The results

```
{ "CUSTOMER" : { "company name" : "Alfreds Futterkiste" } }
{ "CUSTOMER" : { "company name" : "Ana Trujillo Emparedados y helados" } }
{ "CUSTOMER" : { "company name" : "Antonio Moreno Taquería" } }
  ...            ...         ...
```

Select company name of each customer and skip document identifier and remove nestings

aggregate($project)

```
db.orders.aggregate([ {$project:{"Company":"$CUSTOMER.company name","_id":0}} ])
```

The results

```
{ "Company" : "Alfreds Futterkiste" }
{ "Company" : "Ana Trujillo Emparedados y helados" }
{ "Company" : "Antonio Moreno Taquería" }
  ...            ...         ...
```

$project

"$keyname" syntax is used to refer to a value associated with a key "keyname" in the aggregation framework

Select customer addresses and remove nestings

aggregate($project)

```
db.orders.aggregate([ {$project:{"Customer address":"$CUSTOMER.address","_id":0}} ])
```

The results

```
{ "Customer address" : "Obere Str. 57" }
{ "Customer address" : "Avda. de la Constitución 2222" }
{ "Customer address" : "Mataderos  2312" }
  ...           ...         ...
```

Select a name of each customer and concatenate it with its code

aggregate($project)

```
db.orders.aggregate([ {$project:{"Customer address":{$concat:["$CUSTOMER.address",
                                                              "-",
                                                              "$CUSTOMER.customer code"]}}}])
```

The results

```
{ "_id" : "ALFKI", "Customer address" : "Johan Strauss Str. 23-ALFKI" }
{ "_id" : "ANATR", "Customer address" : "Avda. de la Constitución 2222-ANATR" }
{ "_id" : "ANTON", "Customer address" : "Mataderos  2312-ANTON" }
  ...           ...         ...
```

MongoDB Aggregation Framework

Outline

$match

Operator `$match` selects the documents that satisfy a given condition

Find suppliers located in a city `Sandvika`

aggregate($match)

```
db.orders.aggregate([{$match:{"SUPPLIER.city":"Sandvika"}}])
```

Find suppliers located in a city `Sandvika` and display company name, city, and the names of product supplied

aggregate($match,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER.city":"Sandvika"}},
                     {$project:{"SUPPLIER.company name":1,"SUPPLIER.city":1,
                                "SUPPLIER.supplies.PRODUCT.product name":1,"_id":0}}])
```

Find suppliers located in `Germany` supplying a product `Rossle Sauerkraut` and display company name, city, and the names of product supplied

aggregate($match,$match,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER.country":"Germany"}},
                     {$match:{"SUPPLIER.supplies.PRODUCT.product name":"Rossle Sauerkraut"}},
                     {$project:{"SUPPLIER.company name":1,"SUPPLIER.city":1,
                                "SUPPLIER.supplies.PRODUCT.product name":1,"_id":0}}])
```

$match

Find suppliers located in a city `Sandvika` and display company name, city, and the names of product supplied

Incorrect implementation of projection

aggregate($match,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER.city":"Sandvika"}},
                     {$project:{"SUPPLIER.company name":1}},
                     {$project:{"SUPPLIER.city":1}},
                     {$project:{"SUPPLIER.supplies.PRODUCT.product name":1,"_id":0}}])
```

The results

```
{ "SUPPLIER" : {  } })
```

MongoDB Aggregation Framework

Outline

$limit, $skip and $sample

Operation `$limit` passes a given number of documents through a pipeline

Operation `$skip` eliminates a given number of documents from a pipeline

Operation `$sample` randomly picks a given number of documents from a pipeline

Find the first two suppliers

aggregate($match,$limit)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$limit:2}])
```

Find all suppliers except the first two

aggregate($match,$skip)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$skip:2}])
```

$limit, $skip and $sample

Find the third and the fourth supplier

aggregate($match,$limit,$skip)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$limit:4},
                     {$skip:2}])
```

aggregate($match,$skip,$limit)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$skip:2},
                     {$limit:2}])
```

Find all customers located in France and list the sample 2 documents

aggregate($match,$sample)

```
db.orders.aggregate([{$match:{"CUSTOMER.country":"France"}},
                     {$sample:{size:2}}])
```

MongoDB Aggregation Framework

Outline

$count

Operation `$count` counts the total number of documents in a pipeline

List the total number of documents in a collection `orders`

aggregate($count)

```
db.orders.aggregate([{$count:"Total documents"}])
```

A result of counting

```
{ "Total documents" : 117 }
```

Find the total number of suppliers located in Canada

aggregate($match,$count)

```
db.orders.aggregate([{$match:{"SUPPLIER.country":"Canada"}},
                     {$count:"Total suppliers in Canada"}]))
```

A result of counting

```
{ "Total suppliers in Canada" : 2 }
```

$sort

Operation `$sort` sorts the documents

Display the names of companies of all suppliers located in `Canada` and sort the names in ascending order

aggregate($match,$project,$sort)

```
db.orders.aggregate([{$match:{"SUPPLIER.country":"Canada"}},
                     {$project:{"SUPPLIER.company name":1,"_id":0}},
                     {$sort:{"SUPPLIER.company name":1}}])
```

Why the following solution is worse than the one above ?

aggregate($sort,$match,$project)

```
db.orders.aggregate([{$sort:{"SUPPLIER.company name":1}},
                     {$match:{"SUPPLIER.country":"Canada"}},
                     {$project:{"SUPPLIER.company name":1,"_id":0}} ]))
```

Find a company name and postal code of supplier that has the largest postal code

aggregate($match,$project,$sort)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$project:{"SUPPLIER.company name":1,"SUPPLIER.postal code":1,"_id":0}},
                     {$sort:{"SUPPLIER.postal code":-1}},
                     {$limit:1}]))
```

MongoDB Aggregation Framework

Outline

$unwind

Operation `$unwind` creates a separate document for each element of a given array

A document is replicated for each element of an array, i.e. an array is unnested

List the names of products supplied by the first 2 suppliers

aggregate($match,$limit,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$limit:2},
                     {$project:{"SUPPLIER.supplies.PRODUCT.product name":1,"_id":0}}])
```

The results

```
{ "SUPPLIER" : { "supplies" : [ { "PRODUCT" : { "product name" : "Côte de Blaye" } } ] } }
{ "SUPPLIER" : { "supplies" : [ { "PRODUCT" : { "product name" : "Sasquatch Ale" } },
                                { "PRODUCT" : { "product name" : "Steeleye Stout" } },
                                { "PRODUCT" : { "product name" : "Laughing Lumberjack Lager" } } ] } }
```

$unwind

List the names of products supplied by the first 2 suppliers

aggregate($match,$limit,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$limit:2},
                     {$project:{"product":"$SUPPLIER.supplies.PRODUCT.product name","_id":0}}])
```

The results

```
{ "product" : [ "Côte de Blaye" ] }
{ "product" : [ "Sasquatch Ale", "Steeleye Stout", "Laughing Lumberjack Lager" ] }
```

aggregate($match,$limit,$unwind,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$limit:2},
                     {$unwind:"$SUPPLIER.supplies"},
                     {$project:{"product":"$SUPPLIER.supplies.PRODUCT.product name","_id":0}}])
```

The results

```
{ "product" : "Côte de Blaye" }
{ "product" : "Sasquatch Ale" }
{ "product" : "Steeleye Stout" }
{ "product" : "Laughing Lumberjack Lager" }
```

$unwind

List the names of suppliers (companies) and the names of products supplied by the first 2 suppliers

aggregate($match,$limit,$unwind,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$limit:2},
                     {$project:{"supplier":"$SUPPLIER.company name",
                                "product":"$SUPPLIER.supplies.PRODUCT.product name",
                                "_id":0}}])
```

The results

```
{ "supplier" : "Aux joyeux ecclesiastiques", "product" : [ "Côte de Blaye" ] }
{ "supplier" : "Bigfoot Breweries", "product" : [ "Sasquatch Ale", "Steeleye Stout", "Laughing Lumberjack Lager" ]
```

aggregate($match,$limit,$unwind,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$limit:2},
                     {$unwind:"$SUPPLIER.supplies"},
                     {$project:{"supplier":"$SUPPLIER.company name",
                                "product":"$SUPPLIER.supplies.PRODUCT.product name",
                                "_id":0}}])
```

The results

```
{ "supplier" : "Aux joyeux ecclesiastiques", "product" : "Côte de Blaye" }
{ "supplier" : "Bigfoot Breweries", "product" : "Sasquatch Ale" }
{ "supplier" : "Bigfoot Breweries", "product" : "Steeleye Stout" }
{ "supplier" : "Bigfoot Breweries", "product" : "Laughing Lumberjack Lager" }
```

$unwind

List the total number of products supplied by the first 2 suppliers

aggregate($match,$limit,$unwind,$project,$count)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$limit:2},
                     {$unwind:"$SUPPLIER.supplies"},
                     {$project:{"product":"$SUPPLIER.supplies.PRODUCT.product name","_id":0}},
                     {$count:"Total number of products supplied by the first 2 suppliers"}])
```

The results

```
{ "Total number of products supplied by the first 2 suppliers" : 4 }
```

MongoDB Aggregation Framework

Outline

$group

Operation `$group` groups the documents and applies the aggregation functions to each group

Find the total number of suppliers

aggregate($match,$count)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$count:"total"}])
```

aggregate($match,$group)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$group:{"_id": null,"total":{$sum:1}}}])}
```

Find the names of countries, the suppliers come from

aggregate($match,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$project:{"country":"$SUPPLIER.country","_id":0}}])
```

Find the distinct names of countries, the suppliers come from

aggregate($match,$project$,$group)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$project:{"country":"$SUPPLIER.country","_id":0}},
                     {$group:{"_id":"$country"}}]))
```

$group

Find distinct country names together with the total number of suppliers per country

aggregate($match,$group)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$group:{"_id":"$SUPPLIER.country","total":{$sum:1}}}])
```

The results

```
{ "_id" : "Italy", "total" : 2 }
{ "_id" : "Sweden", "total" : 1 }
{ "_id" : "Germany", "total" : 3 }
  ...            ...         ...
```

Find distinct country names together with the total number of suppliers per country

aggregate($match,$project,$group)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$project:{"country":"$SUPPLIER.country","_id":0}},
                     {$group:{"_id":"$country","total":{$sum:1}}}])
```

The results

```
{ "_id" : "Italy", "total" : 2 }
{ "_id" : "Sweden", "total" : 1 }
{ "_id" : "Germany", "total" : 3 }
  ...            ...         ...
```

$group

Find the total number of products supplied by each supplier

aggregate($match,$unwind,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$unwind:"$SUPPLIER.supplies"},
                     {$project:{"supplier":"$SUPPLIER.company name",
                                "product":"$SUPPLIER.supplies.PRODUCT.product name",
                                "_id":0}} ])
```

The results

```
{ "supplier" : "Aux joyeux ecclesiastiques", "product" : "Côte de Blaye" }
{ "supplier" : "Bigfoot Breweries", "product" : "Sasquatch Ale" }
{ "supplier" : "Bigfoot Breweries", "product" : "Steeleye Stout" }
{ "supplier" : "Bigfoot Breweries", "product" : "Laughing Lumberjack Lager" }
  ...            ...         ...
```

aggregate($match,$unwind,$project,$group)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$unwind:"$SUPPLIER.supplies"},
                     {$project:{"supplier":"$SUPPLIER.company name",
                                "product":"$SUPPLIER.supplies.PRODUCT.product name",
                                "_id":0}},
                     {$group:{"_id":"$supplier","total":{$sum:1}}}])
```

The results

```
{ "_id" : "Aux joyeux ecclesiastiques", "total" : 1 }
{ "_id" : "Bigfoot Breweries", "total" : 3 }
  ...            ...         ...
```

$group

Find the total prices of products per supplier

aggregate($match,$unwind,$project)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$unwind:"$SUPPLIER.supplies"},
                     {$project:{"supplier":"$SUPPLIER.company name",
                                "price":"$SUPPLIER.supplies.PRODUCT.unit price",
                                "_id":0}} ])
```

The results

```
{ "supplier" : "Aux joyeux ecclesiastiques", "price" : 263.5 }
{ "supplier" : "Bigfoot Breweries", "price" : 14 }
{ "supplier" : "Bigfoot Breweries", "price" : 18 }
{ "supplier" : "Bigfoot Breweries", "price" : 14 }
  ...            ...         ...
```

aggregate($match,$unwind,$project,$group)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$unwind:"$SUPPLIER.supplies"},
                     {$project:{"supplier":"$SUPPLIER.company name",
                                "product":"$SUPPLIER.supplies.PRODUCT.product name",
                                "price":"$SUPPLIER.supplies.PRODUCT.unit price",
                                "_id":0}},
                     {$group:{"_id":"$supplier","total":{$sum:"$price"}}}])
```

The results

```
{ "_id" : "Aux joyeux ecclesiastiques", "total" : 263.5 }
{ "_id" : "Bigfoot Breweries", "total" : 46 }
  ...            ...         ...
```

$group

Find the averages of all prices of products per supplier

aggregate($match,$unwind,$project,$group)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$unwind:"$SUPPLIER.supplies"},
                     {$project:{"supplier":"$SUPPLIER.company name",
                                "price":"$SUPPLIER.supplies.PRODUCT.unit price",
                                "_id":0}},
                     {$group:{"_id":"$supplier","average":{$avg:"$price"}}}])
```

The results

```
{ "_id" : "Aux joyeux ecclesiastiques", "average" : 263.5 }
{ "_id" : "Bigfoot Breweries", "average" : 15.333333333333334}
  ...            ...         ...
```

$group

Find the minimum and maximum prices of products per supplier

aggregate($match,$unwind,$project,$group)

```
db.orders.aggregate([{$match:{"SUPPLIER":{$exists:true}}},
                     {$unwind:"$SUPPLIER.supplies"},
                     {$project:{"supplier":"$SUPPLIER.company name",
                                "price":"$SUPPLIER.supplies.PRODUCT.unit price",
                                "_id":0}},
                     {$group:{"_id":"$supplier","min":{$min:"$price"},"max":{$max:"$price"}}}])
```

The results

```
{ "_id" : "Aux joyeux ecclesiastiques", "min" : 263.5, "max" : 263.5 }
{ "_id" : "Bigfoot Breweries", "min" : 14, "max" : 18 }
  ...            ...         ...
```

MongoDB Aggregation Framework

Outline

$out

Operation `$out` saves the results of processing in a collection

Find the customer code and the names of products purchased by a customer with a customer code `AROUT` and save the result in a collection `"purchases"`

aggregate($match,$project)

```
db.orders.aggregate([{$match:{"CUSTOMER.customer code":"AROUT"}},
                     {$project:{"Code":"$CUSTOMER.customer code",
                       "Purchased":"$CUSTOMER.submits.ORDER.consists of.ORDER DETAIL.product name"}} ])
```

The results

```
{ "_id" : "AROUT","Code" : "AROUT", "Purchased" : [
                                                    [
                                                      "Gorgonzola Telino",
                                                      "Grandma's Boysenberry Spread",
                                                      "Konbu",
                                                      "Mozzarella di Giovanni",
                                                      "Tofu"
                                                    ]
                                                  ]
}
```

$out

Find the customer code and the names of products purchased by a customer with a customer code `AROUT` and save the result in a collection `"purchases"`

aggregate($match,$project,$unwind)

```
db.orders.aggregate([{$match:{"CUSTOMER.customer code":"AROUT"}},
                     {$project:{"Code":"$CUSTOMER.customer code",
                          "Purchased":"$CUSTOMER.submits.ORDER.consists of.ORDER DETAIL.product name"}},
                     {$unwind:"$Purchased"}])
```

The results

```
{ "_id" : "AROUT","Code" : "AROUT", "Purchased" : [
                                                    "Gorgonzola Telino",
                                                    "Grandma's Boysenberry Spread",
                                                    "Konbu",
                                                    "Mozzarella di Giovanni",
                                                    "Tofu"
                                                  ]
}
```

$out

Find the customer code and the names of products purchased by a customer with a customer code `AROUT` and save the result in a collection `"purchases"`

aggregate($match,$project,$unwind,$unwind)

```
db.orders.aggregate([{$match:{"CUSTOMER.customer code":"AROUT"}},
                     {$project:{"Code":"$CUSTOMER.customer code",
                          "Purchased":"$CUSTOMER.submits.ORDER.consists of.ORDER DETAIL.product name"}},
                     {$unwind:"$Purchased"},
                     {$unwind:"$Purchased"}])
```

The results

```
{ "_id" : "AROUT", "Code" : "AROUT", "Purchased" : "Gorgonzola Telino" }
{ "_id" : "AROUT", "Code" : "AROUT", "Purchased" : "Grandma's Boysenberry Spread" }
{ "_id" : "AROUT", "Code" : "AROUT", "Purchased" : "Konbu" }
{ "_id" : "AROUT", "Code" : "AROUT", "Purchased" : "Mozzarella di Giovanni" }
{ "_id" : "AROUT", "Code" : "AROUT", "Purchased" : "Tofu" }
```

$match,$project,$unwind,$unwind,$out

```
db.orders.aggregate([{$match:{"CUSTOMER.customer code":"AROUT"}},
                     {$project:{"Code":"$CUSTOMER.customer code",
                          "Purchased":"$CUSTOMER.submits.ORDER.consists of.ORDER DETAIL.product name"}},
                     {$unwind:"$Purchased"},
                     {$unwind:"$Purchased"},
                     {$project:{"_id":0}},
                     {$out:"purchases"}])
```

find()

```
db.purchases.find().pretty()
```

<

$out

Find the names, categories, and unit prices of all products supplied and save the result in a collection "products"

aggregate($project)

```
db.orders.aggregate([{$project:{"product":"$SUPPLIER.supplies.PRODUCT",
                                "_id":0 }} ])
```

The results

```
{ "product" : [ { "product name" : "Chef Anton's Cajun Seasoning",
                  "category name" : "Condiments",
                  "quantity per unit" : "48 - 6 oz jars",
                  "unit price" : 25,
                  "units in stock" : 53,
                  "units on order" : 0,
                  "reorder level" : 0,
                  "discontinued" : "N"
                },
  ...             ...         ...
```

aggregate($project,$unwind)

```
db.orders.aggregate([{$project:{"product":"$SUPPLIER.supplies.PRODUCT",
                                "_id":0 }},
                     {$unwind:"$product"} ])
```

$out

Find the names, categories, and unit prices of all products supplied and save the result in a collection `"products"`

aggregate($project,$unwind,$project)

```
db.orders.aggregate([{$project:{"product":"$SUPPLIER.supplies.PRODUCT",
                                "_id":0 }},
                     {$unwind:"$product"},
                     {$project:{"name":"$product.product name",
                                "category":"$product.category name",
                                "price":"$product.unit price"}} ])
```

The results

```
{ "name" : "Côte de Blaye", "category" : "Beverages", "price" : 263.5 }
{ "name" : "Sasquatch Ale", "category" : "Beverages", "price" : 14 }
{ "name" : "Steeleye Stout", "category" : "Beverages", "price" : 18 }
  ...           ...          ...
```

aggregate($project,$unwind,$project,$out)

```
db.orders.aggregate([{$project:{"product":"$SUPPLIER.supplies.PRODUCT",
                                "_id":0 }},
                     {$unwind:"$product"},
                     {$project:{"name":"$product.product name",
                                "category":"$product.category name",
                                "price":"$product.unit price"}},
                     {$out:"products"}])
```

find()

```
db.products.find().pretty()
```

TOP

MongoDB Aggregation Framework

Outline

$lookup

Operation `$lookup` performs a left outer join to an unsharded collection in the same database to filter in documents from the "joined" collection for processing

A collection `purchases` created earlier

A collection purchases

```
{ "_id" : ObjectId("61342804f26c5a766e0e1c6f"),
  "Code" : "AROUT", "Purchased" : "Gorgonzola Telino" }
{ "_id" : ObjectId("61342804f26c5a766e0e1c70"),
  "Code" : "AROUT", "Purchased" : "Grandma's Boysenberry Spread" }
  ...          ...        ...
```

A collection `products` created earlier

A collection products

```
{ "_id" : ObjectId("61342773f26c5a766e0e1c1a"),
  "name" : "Gorgonzola Telino","category" : "Dairy Products","price" : 12.5 }
{ "_id" : ObjectId("61342773f26c5a766e0e1c22"),
  "name" : "Grandma's Boysenberry Spread","category" : "Condiments","price" : 25 }
  ...          ...        ...
```

$lookup

A left outer join of a collection `purchases` with a collection `products` over a condition `purchases.Purchased = products.name`

aggregate($lookup)

```
db.purchases.aggregate([{$lookup: {from:"products",
                                   localField:"Purchased",
                                   foreignField:"name",
                                   as:"result"}} ])
```

purchases left outer join products

```
{ "_id" : ObjectId("61342804f26c5a766e0e1c6f"),
  "Code" : "AROUT","Purchased" : "Gorgonzola Telino",
   "result" : [ {"_id" : ObjectId("61342773f26c5a766e0e1c1a"),
                 "name" : "Gorgonzola Telino",
                 "category" : "Dairy Products",
                 "price" : 12.5}
              ]
}
{ "_id" : ObjectId("61342804f26c5a766e0e1c70"),
  "Code" : "AROUT","Purchased" : "Grandma's Boysenberry Spread",
  "result" : [ { "_id" : ObjectId("61342773f26c5a766e0e1c22"),
                 "name" : "Grandma's Boysenberry Spread",
                 "category" : "Condiments",
                 "price" : 25}
             ]
}
  ...          ...        ...
```

$lookup

A left outer join of a collection `products` with a collection `purchases` over a condition `products.name = purchases.Purchased`

aggregate($lookup)

```
db.products.aggregate([{$lookup: {from:"purchases",
                                  localField:"name",
                                  foreignField:"Purchased",
                                  as:"result"}} ])
```

products left outer join purchases

```
{"_id" : ObjectId("61342773f26c5a766e0e1c22"), "name" : "Grandma's Boysenberry Spread","category" : "Condiments", "price" : 25,
 "result" : [{"_id" : ObjectId("61342804f26c5a766e0e1c70"),"Code" : "AROUT","Purchased" : "Grandma's Boysenberry Spread"}]
}
{"_id" : ObjectId("61342773f26c5a766e0e1c1a"),"name" : "Gorgonzola Telino","category" : "Dairy Products","price" : 12.5,
 "result" : [{"_id" : ObjectId("61342804f26c5a766e0e1c6f"),"Code" : "AROUT","Purchased" : "Gorgonzola Telino"}]
}
  ...          ...        ...

{"_id" : ObjectId("61342773f26c5a766e0e1c0e"),"name" : "Côte de Blaye","category" : "Beverages","price" : 263.5,
 "result" : [ ]
}
{"_id" : ObjectId("61342773f26c5a766e0e1c0f"),"name" : "Sasquatch Ale","category" : "Beverages","price" : 14,
 "result" : [ ]
}
  ...          ...        ...
```

References

MongoDB Manual, Aggregation

Banker K., Bakkum P., Verch S., Garret D., Hawkins T., MongoDB in Action, 2nd ed., Manning Publishers, 2016

Chodorow K. MongoDB The Definitive Guide, O'Reilly, 2013