NoSQL Databases

PL05 – Aggregation and Indexing in MongoDB

Teacher: Cristiana Neto
Email: cristiana.neto@algoritmi.uminho.pt

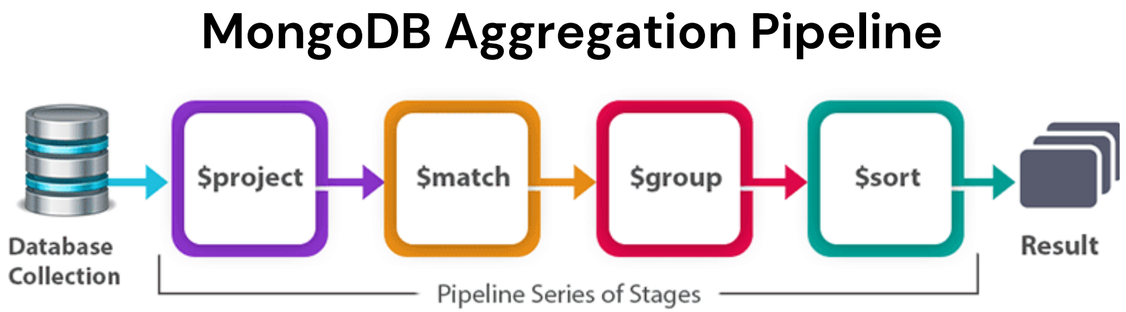
Office hours:
Friday 10h-11h

Summary

Aggregation

Pipeline

Aggregation operations in MongoDB are operations that group data together and return computed results, such as sums and counts.



`$project` – Select the document fields we want to work with

`$match` – It filters the documents we need to work with, which suit our needs

`$group` – does the aggregation work

`$sort` – sorts the resulting documents the way we want (descending or ascending)

Aggregation

Aggregation Syntax

```
db.collectionName.aggregate(pipeline, options)
```

- *collectionName* – Collection Name
- *pipeline* – Array that contains the steps
- *options* – Optional parameters for aggregation

```
pipeline = [
        { $match : { … } },
        { $group : { … } },
        { $sort : { … } }
       ]
```

Aggregation

Example #1

Create db:

```
> use pl05
```

Create Universities Collection and Add Documents:

```
> db.universities.insertMany([ {  country : 'Spain',  city : 'Salamanca',  name : 'USAL',  location : {    type : 'Point',    coordinates : [ -5.6722512,17, 40.9607792 ]  },  students : [    { year : 2014, number : 24774 },    { year : 2015, number : 23166 },    { year : 2016, number : 21913 },    { year : 2017, number : 21715 }  ] }, {  country : 'Spain',  city : 'Salamanca',  name : 'UPSA',  location : {    type : 'Point',    coordinates : [ -5.6691191,17, 40.9631732 ]  },  students : [    { year : 2014, number : 4788 },    { year : 2015, number : 4821 },    { year : 2016, number : 6550 },    { year : 2017, number : 6125 }  ] } ])
```

Create courses collection and add documents:

```
> db.courses.insertMany([ {  university : 'USAL',  name : 'Computer Science',  level : 'Excellent' }, {  university : 'USAL',  name : 'Electronics',  level : 'Intermediate' }, {  university : 'USAL',  name : 'Communication',  level : 'Excellent' } ])
```

Aggregation

➡ Example #1

3. Save the results of an aggregation to a new collection:

```
db.universities.aggregate([
  { $group: { _id: '$name', totaldocs: { $sum: 1 } } },
  { $out: 'aggResults' }
])
```

4. Obtain a document for each element of the USAL university student array:

```
db.universities.aggregate([
     { $match : { name : 'USAL' } },
     { $unwind : '$students' } ])
```

Aggregation

Example #1

5. Sort the documents obtained in the previous point by the number of students in descending order, projecting only the year and the number of students:

```
db.universities.aggregate([
      { $match : { name : 'USAL' } },
      { $unwind : '$students' },
      { $project : { _id : 0, 'students.year' : 1, 'students.number' : 1 } },
      { $sort : { 'students.number' : -1 } } ])
```

Aggregation

Example #1

6. Limit Search:

```
db.universities.aggregate([
      { $match : { name : 'USAL' } },
      { $unwind : '$students' },
      { $project : { _id : 0, 'students.year' : 1, 'students.number' : 1 } },
      { $sort : { 'students.number' : -1 } },
      { $limit : 1 }
])
```

Aggregation

Example #1

7. Check the amount of documents obtained in the output of the previous steps of the pipeline:

```
db.universities.aggregate([
      { $unwind : '$students' },
      { $count : 'total_documents' } ])
```

Aggregation

➡ Example #1

8. Merge fields from two collections:

```
db.universities.aggregate([
      { $match : { name : 'USAL' } },
      { $project : { _id : 0, name : 1 } },
      { $lookup :
            { from : 'courses',
              localField : 'name',
              foreignField : 'university',
              as : 'courses' }
      } ])
```

PS: If we want this query to run quickly, we will need to index the `name` field in the `universities` collection and the `university` field in the `courses` collection.

Aggregation

Example #1

9. Total number of students who have belonged to each of the universities in descending order:

```
db.universities.aggregate([
      { $unwind : '$students' },
      { $group : { _id : '$name', totalalumni : { $sum : '$students.number' } } },
      { $sort : { totalalumni : -1 } }
])
```

Aggregation

Example #2

Create db:

```
> use pizza
```

Create order collection and add documents:

```
> db.orders.insertMany( [
{_id: 0, name: "Pepperoni", size: "small", price: 19, quantity: 10, date: ISODate("2021-03-13T08:14:30Z")},
{_id: 1, name: "Pepperoni", size: "medium", price: 20, quantity: 20, date: ISODate("2021-03-13T09:13:24Z")},
{_id: 2, name: "Pepperoni", size: "large", price: 21, quantity: 30, date: ISODate("2021-03-17T09:22:12Z")},
{_id: 3, name: "Cheese", size: "small", price: 12, quantity: 15, date: ISODate("2021-03-13T11:21:39.736Z")},
{_id: 4, name: "Cheese", size: "medium", price: 13, quantity:50, date: ISODate("2022-01-12T21:23:13.331Z")},
{_id: 5, name: "Cheese", size: "large", price: 14, quantity: 10, date: ISODate("2022-01-12T05:08:13Z")},
{_id: 6, name: "Vegan", size: "small", price: 17, quantity: 10, date: ISODate("2021-01-13T05:08:13Z")},
{_id: 7, name: "Vegan", size: "medium", price: 18, quantity: 10, date: ISODate("2021-01-13T05:10:13Z")}
])
```

Aggregation

Example #2

1. Return the total order quantity of medium pizzas grouped by pizza name:

```
> db.orders.aggregate( [
      // Stage 1: Filter your order documents by pizza size
      {
         $match: { size: "medium" }
      },
      // Stage 2: Group the remaining documents by the name of the pie and calculate
the total quantity
      {
         $group: { _id: "$name", totalQuantity: { $sum: "$quantity" } }
      }
] )
```

Result:

```
[
 { _id: 'Cheese', totalQuantity: 50 },
 { _id: 'Vegan', totalQuantity: 10 },
 { _id: 'Pepperoni', totalQuantity: 20 }
]
```

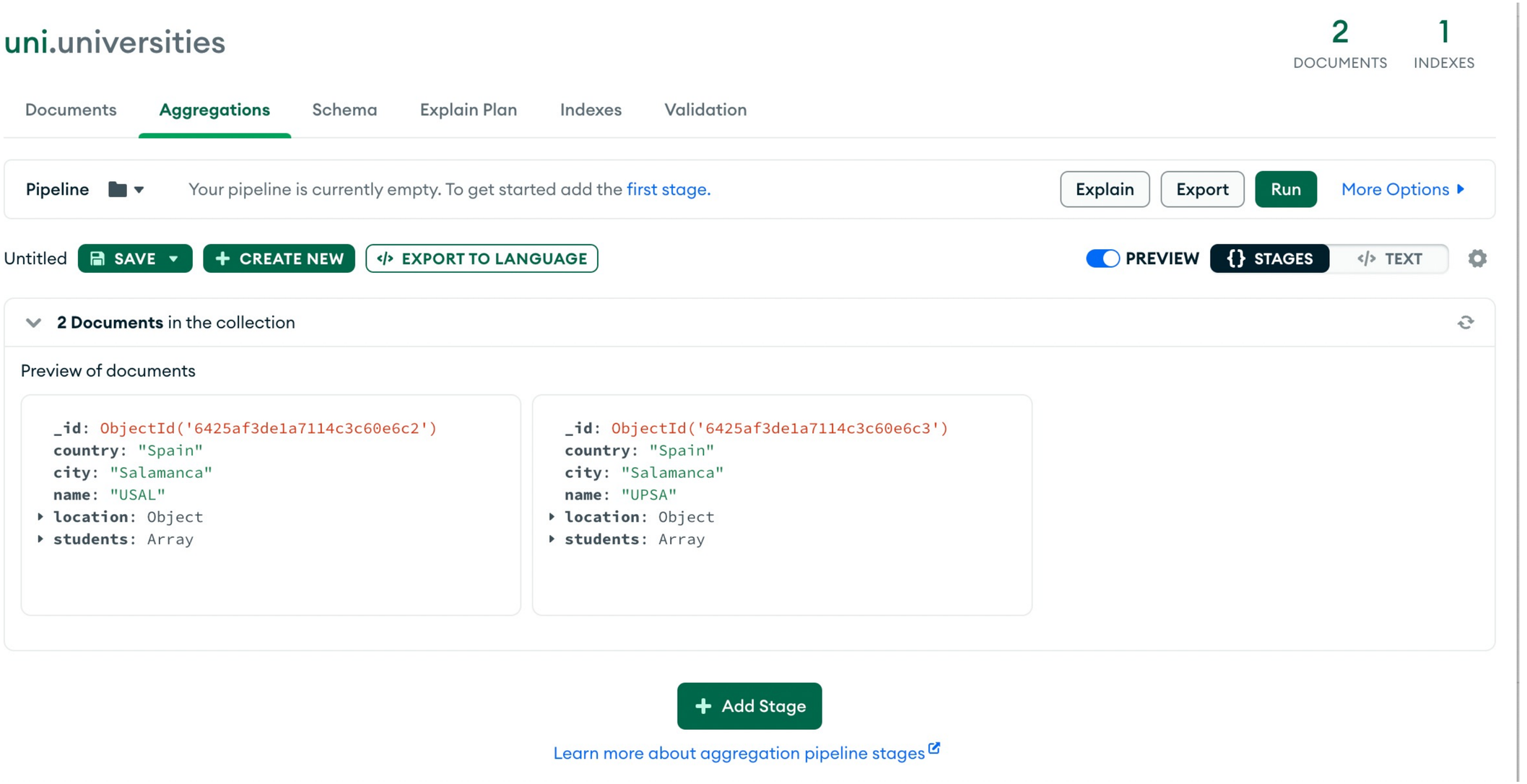
Aggregation

Example #2

2. Return the total value and average order quantity between two dates:

```
> db.orders.aggregate( [
      // Stage 1: Filter order documents by dates
      {
         $match: {"date": {$gte: new ISODate("2020-01-30"), $lt: new ISODate("2022-01-30")}}
      },
      // Stage 2: Group the remaining documents by date and calculate the results
      {
         $group: {_id: {$dateToString: {format: "%Y-%m-%d", date: "$date" } },
           totalOrderValue: {$sum: {$multiply: ["$price", "$quantity"] } },
           averageOrderQuantity: {$avg: "$quantity" }}},
      // Stage 3: Sort documents by totalOrderValue in descending order
      {
         $sort: {totalOrderValue: -1}}
])
```

Aggregation in Compass



Indexing

Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must perform a *collection scan*, i.e. scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

```
db.collection.createIndex( <key and index type specification>, <options> )
```

There are several types of indexes in MongoDB, including simple, compound, text, and geospatial indexes. Simple indexes index a single field, while compound indexes index multiple fields together. Text indexes are used for full-text search queries, while geospatial indexes are used for location-based queries.

FE04 – Aggregation in MongoDB

Universidade do Minho
Departamento de Informática

FE04

Curso: Mestrado em Informática / Mestrado em Bioinformática
U.C.: Bases de Dados NoSQL

Ficha de Exercícios 04 - PL05	
Docente:	António Abelha / Cristiana Neto
Tema:	Agregação no MongoDB
Turma:	Prática Laboratorial
Ano Letivo:	2023-2024 – 2º Semestre
Duração da aula:	2 horas

NoSQL Databases

PL05 – Aggregation and Indexing in MongoDB

Teacher: Cristiana Neto
Email: cristiana.neto@algoritmi.uminho.pt

Office hours:
Friday 10h-11h