



mongoDB

MESIIN480323 : No SQL 19/01

PROJECT REPORT
On
MongoDB
reuters.json

GABRIEL CHAIX & MALO LEROY & CHARLES TERRIER

ÉCOLE SUPERIEURE D'INGENIEURS LEONARD DE VINCI

ANNEE 2023 / 2024

INDEX

INTRODUCTION AND PROBLEM STATEMENT	2
I. Data Importation.....	3
II. Data cleaning	4
III. Queries.....	5
A. Simple queries (6).....	5
1. List of two first articles	5
2. List of textes titles sorted by date	5
3. Titles, id and dateline of texts where id "120"	5
4. Count of articles whose place is France.	6
5. Count of publications whose "texts" is null.	6
6. Give the number articles written by each author ?.....	6
B. Complex queries (2)	6
1. How much articles have been published per month?	6
2. For each topic, give the average number of people that wrote on. Sort the result decreasingly.....	7
C. Hard query (1)	8
1. Between 1984 and 2024, how long have been the articles for a certain topic for each country? Sort by average body length.....	8

INTRODUCTION AND PROBLEM STATEMENT

Reuters is a news agency founded in 1851 in London. It is one of the global and generalist news agencies, an activity that represents a part of its revenue, mainly devoted to financial information. It is one of the greatest news agencies in the world. We want to create a MongoDB database with reuters articles and query it in different levels of complexity.

I. Data Importation

In a first manner, we'll setup the docker container with mongo db image and make sure everything is working properly.

```

$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
36a37948228b mongo:latest "docker-entrypoint.s..." About a minute ago Up About a minute 8.0.0.0:27017->27017/tcp hardcore_mayer

$ docker exec -i -t hardcore_mayer bash
root@36a37948228b:/# mongo
Current Mongosh Log ID: 65c53ce1532bd5a04ad6a8e8
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Using MongoDB: 7.0.5
Using Mongosh: 2.1.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy). You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2024-02-08T20:40:57.010+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
2024-02-08T20:40:57.706+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2024-02-08T20:40:57.706+00:00: vm.max_map_count is too low
-----

test>

```

Now we must create our database using python and the pymongo package. To do this, we run the following script to build the database schema and fill the tables with our dataset content. After downloading our [json file](#) put it in the same directory as our code.

```

import json
from pymongo import MongoClient, InsertOne

client = MongoClient("mongodb://localhost:27017")
db = client.reuters
collection = db.articles
requesting = []

with open("MongoDB/reuters.json") as f:
    for jsonObj in f:
        myDict = json.loads(jsonObj)
        requesting.append(InsertOne(myDict))

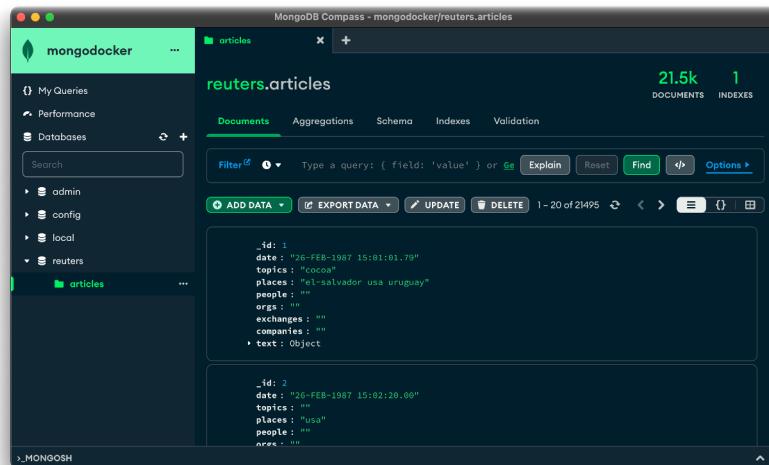
result = collection.bulk_write(requesting)
client.close()

```

Then here is the importation script we've wrote

←

After importing all the rows, we make sure that each row is correct.



Using mongo compass, we can see everything looks alright.

II. Data cleaning

The first step we've taken into cleaning has been to update all date entry to the valid date type.

```
test> show dbs
admin   40.00 KiB
config  108.00 KiB
local   72.00 KiB
reuters 12.38 MiB

test> use reuters
switched to db reuters
reuters> db.articles.find({ _id : 1 })
{
  "_id": 1,
  "date": "2014-03-08T18:02:41.79",
  "topics": "cocoa",
  "places": "vt-selvagens via argenpress",
  "people": [],
  "orgs": [],
  "exchanges": [],
  "companies": [],
  "text": {
    "title": "ARGENTINA COCOA REVIEW",
    "dateline": "SALVADOR, Feb 26 --",
    "body": "Showers continued throughout the week in the Bahia coffee zone, alleviating the drought since early January and improving prospects for the coming temper-  
so, although normal humidity levels have not been restored, Comissaria Smith said in its weekly review. The dry period should the temps will be late this year.  
Arrivals for the week ended February 26 were 193,140 bags of 90 kg, making a cumulative total for the season of 2,129,160 bags. 1,641 of the same stage left  
earlier, leaving 1,468 bags still available for arrival. Comissaria Smith also noted there is still some doubt as to how  
many bags will be available for arrival in the future, due to the current situation with political unrest and also bags that were sent to the  
US. All these are following the last bags left in the hands of farmers, middlemen and processors. There are discussions about the fate of this cargo  
as well as for export as shippers are now experiencing difficulties in obtaining shipping certificates and documents. In view of the lower quality over recent weeks it  
is believed that a good part of this cargo held no commitment. Comissaria Smith said sugar prices rose to $40 to $50 cruzeiros per arroba of 15 kilos. He  
also said shippers were reluctant to offer nearby shipment and only limited sales were booked for March shipment at 1,750 to 1,780 cruzeiros per tonne to ports to be named.  
New crop sales were also light and all to local ports with Jundiai selling at 1,950 and 1,900 dirh and at 31 and 45 dirh under New York 100, August at 1,720, 1,720  
and 1,700 dirh per tonne PGN. Argentine sales of butter were made: March/April sold at 1,450, 1,440, 1,440 and 4,100 dirh. April/May butter went at 1,77 dirh  
per kg. May, June/July at 4,050 and 4,110 dirh. Aug/Sept at 4,250 to 4,280 dirh and at 2,27 and 2,28 times New York 100 and 20/21 times New York  
100, Comissaria Smith said. Payments were the local, convertible currency areas, Uruguay and open ports. Cash sales were registered at 700 to 800 dirh per  
kg. March/April, 100 dirh for May, 700 dirh for Aug and 80 to 90 times New York for Oct/Dec. Buyers were the U.S., Argentina, Uruguay and convertible currency areas.  
Liquor sales were limited with Maranhao selling at 2,370 and 2,380 dirh. June/July at 2,370 dirh and at 1,05 times New York 100, August at 2,400 dirh and
```

This aggregation pipeline transforms documents in a collection by converting the format of the date field to a date object and selectively projects specific fields as well as nested fields within text (such as title, dateline, and body), and the people field, essentially filtering and formatting the document structure for the output.

```
db.articles.aggregate([
  {
    $project: {
      date: {
        $ToDate: "$date"
      },
      places: 1,
      companies: 1,
      topics: 1,
      exchanges: 1,
      _id: 1,
      orgs: 1,
      "text.title": 1,
      "text.dateLine": 1,
      "text.body": 1,
      people: 1
    }
  },
  {
    $merge: {
      into: "articles",
      on: "_id",
      whenMatched: "replace",
      whenNotMatched: "insert"
    }
  }
])
```

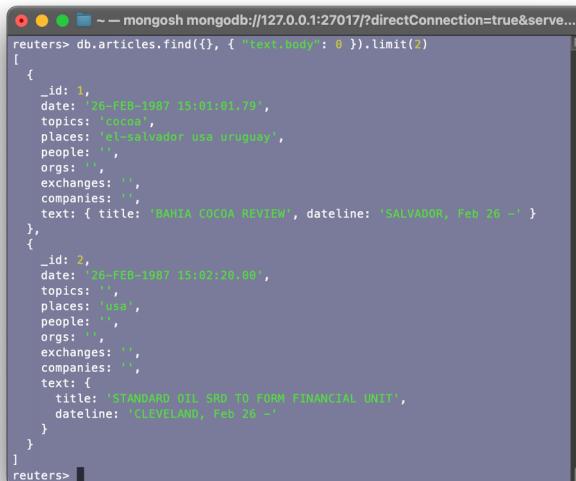


This conversion requires to deal with few strings exception that we've modified using \$set and \$regexMatch statements.

III. Queries

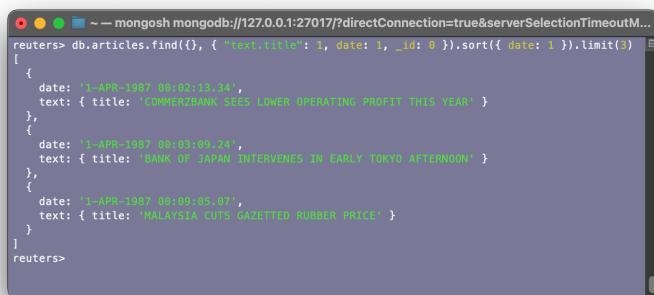
A. Simple queries (6)

1. List of two first articles



```
reuters> db.articles.find({}, { "text.body": 0 }).limit(2)
[ {
    "_id": 1,
    "date": "26-FEB-1987 15:01:01.79",
    "topics": "cocoa",
    "places": "el-salvador usa uruguay",
    "people": "",
    "orgs": "",
    "exchanges": "",
    "companies": "",
    "text": { "title": "NARIA CHOCO REVIEW", "dateline": "SALVADOR, Feb 26 -" }
},
{
    "_id": 2,
    "date": "26-FEB-1987 15:02:18.99",
    "topics": "usa",
    "places": "usa",
    "people": "",
    "orgs": "",
    "exchanges": "",
    "companies": "",
    "text": {
        "title": "STANDARD OIL SAB TO FORM FINANCIAL UNIT",
        "dateline": "CLEVELAND, Feb 26 -"
    }
}
]
reuters>
```

2. List of texts titles sorted by date



```
reuters> db.articles.find({}, { "text.title": 1, "date": 1, "_id": 0 }).sort({ date: 1 }).limit(3)
[ {
    "date": "1 APR 1987 09:03:03.13",
    "text": { "title": "COMMERCIAL BANK SEEKS LOWER OPERATING PROFIT THIS YEAR" }
},
{
    "date": "1 APR 1987 09:03:03.24",
    "text": { "title": "BANK OF JAPAN INTERVENES IN EARLY TOKYO AFTERNOON" }
},
{
    "date": "1 APR 1987 09:03:03.47",
    "text": { "title": "MALAYSIA CUTS GAZETTED RUBBER PRICE" }
}
]
reuters>
```

3. Titles, id and dateline of texts where id "120"



```
reuters> db.articles.find({ "_id": 120 }, { "text.title": 1, "text.dateline": 1, "_id": 1 })
[ {
    "_id": 120,
    "text": {
        "title": "HIGH POINT FINANCIAL CORP SETS OFFERING",
        "dateline": "BRANCHVILLE, N.J., Feb 26 -"
    }
}
]
reuters>
```

4. Count of articles whose place is France.

```
reuters> db.articles.countDocuments({ "place": "france" })
272
reuters>
```

5. Count of publications whose "texts" is null.

```
reuters> db.articles.countDocuments({ "text.body": null })
148
reuters>
```

6. Give the number articles written by each author ?

```
reuters> db.articles.aggregate([
...   {
...     $group: {
...       _id: "$people",
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       people: "$_id",
...       count: 1
...     }
...   }
... ])
[ { count: 3, people: 'poehl kohl' },
{ count: 1, people: 'duisenberg' },
{
  count: 1,
  people: 'stultenberg poehl balladur de-laresiere miyazawa sumita leigh-pemberton gorta james-baker'
},
{ count: 3, people: 'corrigan' },
{ count: 1, people: 'yuutter lyng' },
{ count: 2, people: 'keating hawks' },
{ count: 1, people: 'garcia de-la-madrid manteria-aquayo' },
{ count: 2, people: 'reagan james-baker greenspan sprinkel' },
{ count: 1, people: 'james-baker greenspan' },
{ count: 1, people: 'young' },
{ count: 1, people: 'james-baker nakasone' },
{ count: 3, people: 'james-baker volcker' } ]
```

In this part, we performed a series of queries to get obvious and trivial information about the content of our dataset. Now, we will go further across the following complex queries.

B. Complex queries (2)

1. How much articles have been published per month?

Since we converted the dates to date type, we can simplify the format of dates.

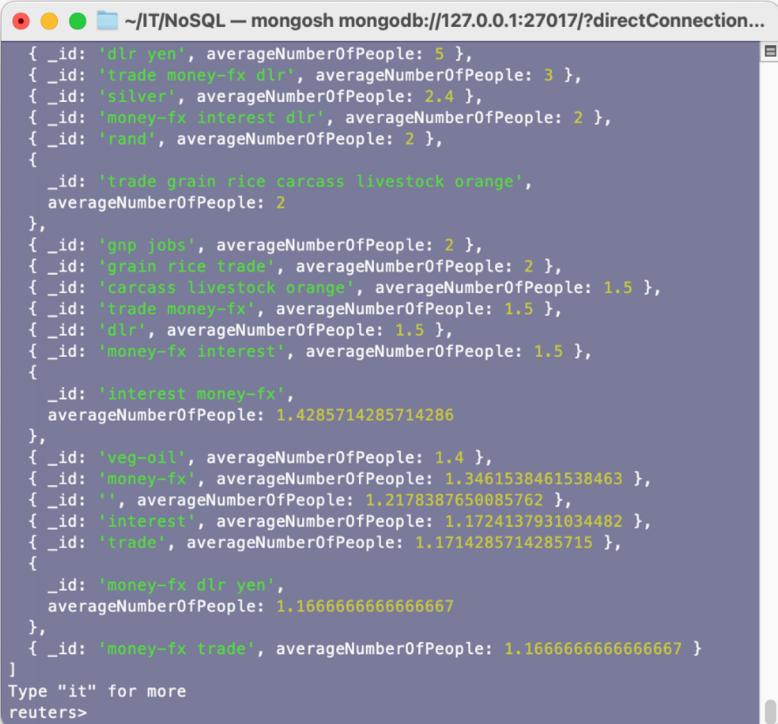


```

reuters> db.articles.aggregate([
...   {
...     $group: {
...       _id: {
...         $dateToString: {
...           format: "%Y-%m",
...           date: "$date"
...         }
...       },
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $sort: { "date": 1 }
...   }
... ])
[
  { _id: '1987-02', count: 228 },
  { _id: '1987-06', count: 2552 },
  { _id: '1987-10', count: 1511 },
  { _id: '1987-03', count: 11543 },
  { _id: '1987-04', count: 5661 }
]
reuters>

```

2. For each topic, give the average number of people that wrote on. Sort the result decreasingly.



```

db.articles.aggregate([
{
  $match: {
    "people": { $exists: true, $ne: "" }
  }
},
{
  $addFields: {
    "peopleArray": { $split: ["$people", " "] }
  }
},
{
  $group: {
    _id: "$topics",
    averageNumberOfPeople: {
      $avg: { $size: "$peopleArray" }
    }
  }
},
{
  $sort: {
    "averageNumberOfPeople": -1
  }
}
])

```

```

{ _id: 'dlr yen', averageNumberOfPeople: 5 },
{ _id: 'trade money-fx dlr', averageNumberOfPeople: 3 },
{ _id: 'silver', averageNumberOfPeople: 2.4 },
{ _id: 'money-fx interest dlr', averageNumberOfPeople: 2 },
{ _id: 'rand', averageNumberOfPeople: 2 },
{
  _id: 'trade grain rice carcass livestock orange',
  averageNumberOfPeople: 2
},
{ _id: 'gno jobs', averageNumberOfPeople: 2 },
{ _id: 'grain rice trade', averageNumberOfPeople: 2 },
{ _id: 'carcass livestock orange', averageNumberOfPeople: 1.5 },
{ _id: 'trade money-fx', averageNumberOfPeople: 1.5 },
{ _id: 'dlr', averageNumberOfPeople: 1.5 },
{ _id: 'money-fx interest', averageNumberOfPeople: 1.5 },
{
  _id: 'interest money-fx',
  averageNumberOfPeople: 1.4285714285714286
},
{ _id: 'veg-oil', averageNumberOfPeople: 1.4 },
{ _id: 'money-fx', averageNumberOfPeople: 1.3461538461538463 },
{ _id: '', averageNumberOfPeople: 1.2178387650085762 },
{ _id: 'interest', averageNumberOfPeople: 1.1724137931034482 },
{ _id: 'trade', averageNumberOfPeople: 1.1714285714285715 },
{
  _id: 'money-fx dlr yen',
  averageNumberOfPeople: 1.1666666666666667
},
{ _id: 'money-fx trade', averageNumberOfPeople: 1.1666666666666667 }
]
Type "it" for more
reuters>

```

C.Hard query (1)

- Between 1984 and 2024, how long have the articles been existing for a certain topic for each country? Sort by average body length.

To answer this question correctly, we make an aggregation pipeline that performs the following operations:

Filters the documents between certain date having not null text. →

```
db.articles.aggregate([
  {
    $match: {
      date: {
        $gte: ISODate("1984-01-01T00:00:00Z"),
        $lt: ISODate("2025-01-01T00:00:00Z")
      },
      "text.body": { $ne: null }
    }
  },
  {
    $project: {
      topics: { $split: ["$topics", " "] },
      places: { $split: ["$places", " "] },
      people: { $split: ["$people", " "] },
      textLength: { $strLenCP: "$text.body" }
    }
  },
  {
    $unwind: "$topics"
  },
  {
    $unwind: "$places"
  },
  {
    $group: {
      _id: {
        topic: "$topics",
        place: "$places"
      },
      averageLength: { $avg: "$textLength" }
    }
  },
  {
    $sort: {
      "averageLength": -1
    }
  }
]);
```

Turns people topics and places to array of strings using split. →

Groups documents by topics, then subgroups by places. →

Calculates the average length of text.body for each subgroup. →

Sorts results by average body length, in descending order. →

According to the results, we see that the longest articles between 1984 and 2024 was talking about cotton-oil in the United States.

```
[{"_id": {"topic": "cotton-oil", "place": "usa"}, "averageLength": 11733}, {"_id": {"topic": "copper", "place": "canada"}, "averageLength": 8409}, {"_id": {"topic": "rubber", "place": "brazil"}, "averageLength": 8409}, {"_id": {"topic": "coffee", "place": "malaysia"}, "averageLength": 8409}, {"_id": {"topic": "coffee", "place": "canada"}, "averageLength": 8409}, {"_id": {"topic": "rubber", "place": "west-germany"}, "averageLength": 8409}, {"_id": {"topic": "rubber", "place": "uk"}, "averageLength": 8409}, {"_id": {"topic": "tin", "place": "canada"}, "averageLength": 8409}, {"_id": {"topic": "sugar", "place": "malaysia"}, "averageLength": 8409}, {"_id": {"topic": "sugar", "place": "guatemala"}, "averageLength": 6309}, {"_id": {"topic": "sugar", "place": "ecuador"}, "averageLength": 6309}, {"_id": {"topic": "tin", "place": "brazil"}, "averageLength": 5598}, {"_id": {"topic": "rubber", "place": "france"}, "averageLength": 5585.5}, {"_id": {"topic": "tin", "place": "france"}, "averageLength": 5588}, {"_id": {"topic": "sugar", "place": "brazil"}, "averageLength": 3416}, {"_id": {"topic": "copper", "place": "oland"}, "averageLength": 3299}, {"_id": {"topic": "tin", "place": "west-germany"}, "averageLength": 3244}, {"_id": {"topic": "tin", "place": "italy"}, "averageLength": 3219}, {"_id": {"topic": "sheep", "place": "malaysia"}, "averageLength": 5171.5}, {"_id": {"topic": "green", "place": "malaysia"}, "averageLength": 5171.5}, {"_id": {"topic": "baudriyal", "place": "uk"}, "averageLength": 4973}], Type "it" for more routers>
```