MongoDB Report

Minka Firth

Fontys Hogeschool
Semester 4 – Data Processing – Module 3
January 2022

Table of Contents

MongoDB choice

I decided to go with a MongoDB database. While I have no experience with NoSQL databases, I do have some experience with JSON files, so I thought maybe this would come in handy (turns out it didn't really).

MongoDB was also one of the few more 'beginner friendly' options available for a document-store type of database. There are a *lot* of tutorials and documentation available these days, but this did not really always prove to be a blessing. There are a lot of tutorials from MongoDB themselves, and they al sort of steer you into a certain direction (using the Atlas browser service). So finding the proper solutions didn't always go smoothly. This was however my main reason for using this system.

About MongoDB

MongoDB is a document-store database, meaning that each record is considered a document. Assigned to each document is a unique key. This key is used to retrieve the documents. These Documents are often stored in a JSON(/-like) format. They are bundled together into *Collections.* There can be multiple collections within one database. As opposed to RDBMs, MongoDB does not require the user to make a schema beforehand, and is therefore often called *Schemaless.*

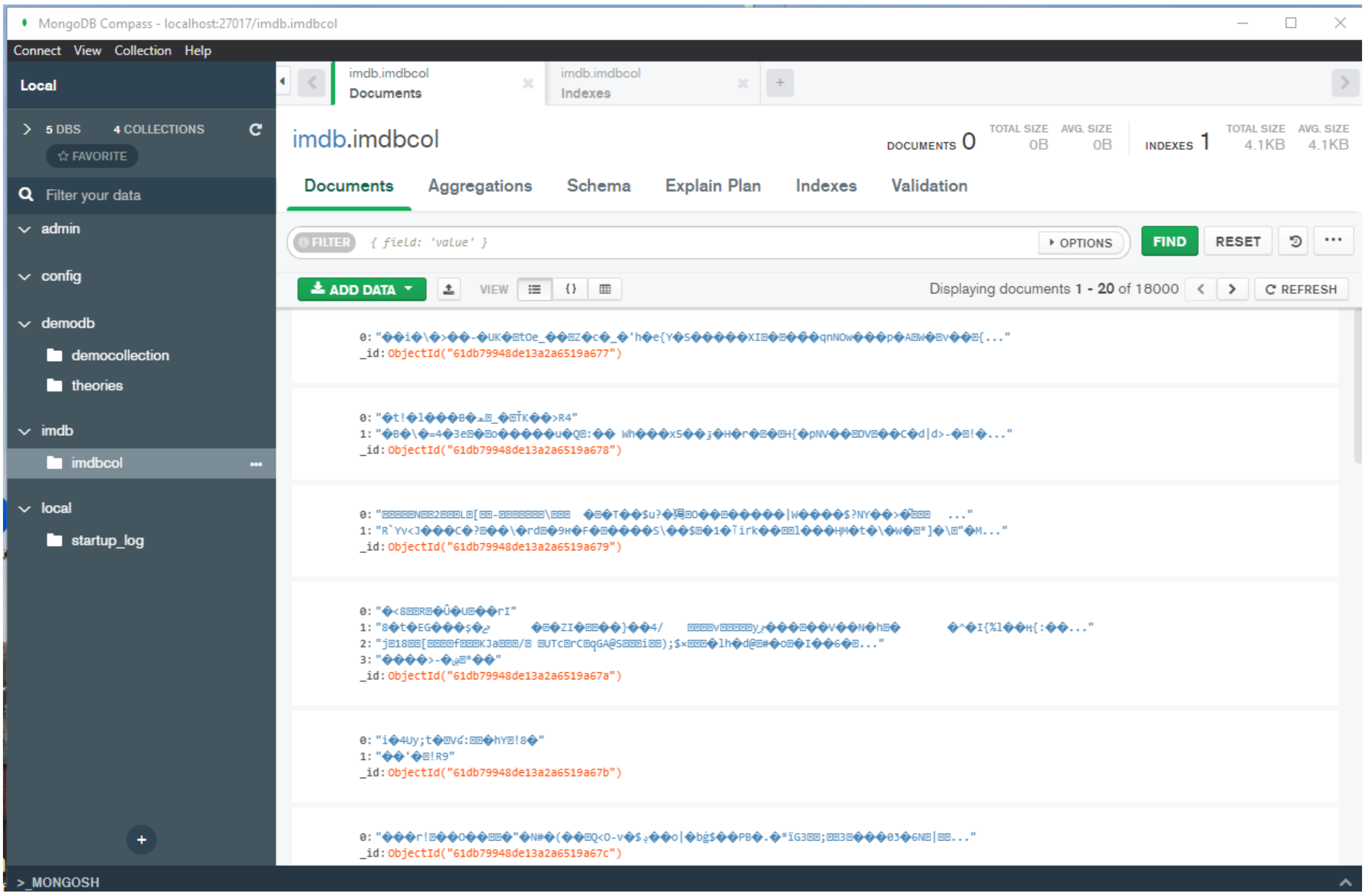
Installation

I didn't really run into any problems, except for wanting to run the server out of an undirected gitbash shell. This however, proved to be impossible. I could only run the server if I opened the shell in the actual folder on my laptop. Since it worked, I thought it was fine, but I mentioned it to my boyfriend, who got all confused. He tried loads of different solutions, trying to add a new path and everything, but none of it worked. I was just glad it *was* working, so I told him not to worry about it (to no avail).

When trying to run the *mongoimport* command to import my bigger TSV-file, I realised I had forgotten to download the MongoDB tools as well.

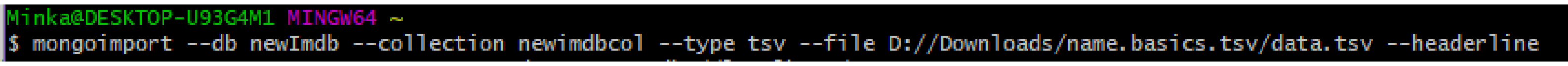
Importing

When I had figured out installing everything, I started out with a small sample dataset concerning Game of Thrones theories. This was a simple JSON file that I could just load really easily into the database using the UI. I tried some queries and didn't run into any problems. I thought, okay nice, let's get started with the real deal. When I tried to import the actual dataset I wanted to work with, the Imdb dataset, I ran into problems. When I wanted to import it via the Mongodb UI, *Compass,* the only the file options available were JSON or CSV. My file turned out to be a TSV file, so I wasn't sure how to proceed. It would let me import the document as a CSV file, but it didn't look too pretty.


MongoDB Compass - localhost:27017/imdb.imdbcol
Connect View Collection Help
Local
5 DBS 4 COLLECTIONS
☆ FAVORITE
Filter your data
admin
config
demodb
democollection
theories
imdb
imdbcol
local
startup_log
imdb.imdbcol Documents
imdb.imdbcol Indexes
imdb.imdbcol
DOCUMENTS 0 TOTAL SIZE 0B AVG. SIZE 0B
INDEXES 1 TOTAL SIZE 4.1KB AVG. SIZE 4.1KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER { field: 'value' }
OPTIONS FIND RESET
ADD DATA VIEW
Displaying documents 1 - 20 of 18000
REFRESH
>_MONGOSH

```
0: "��i�\�>��-�UK�□tOe_��□Z�c�_�'h�e{Y�S�����XI□����qnNOw���p�A□W�□v��□{..."
_id: ObjectId("61db79948de13a2a6519a677")

0: "�t!�1���B�⁎□_�□ŤK��>R4"
1: "�B�\�=4�3e□�□o�����u�Q□:�� Wh���x5��ʝ�H�r�□�□H{�pNV��□DV□��C�d|d>-�□!�..."
_id: ObjectId("61db79948de13a2a6519a678")

0: "□□□□□N□□2□□□L□[□□-□□□□□□□\□□□  �□�T��$u?�□□O��□������|W�����$?NY��>�□□□  ..."
1: "R`Yv<J���C�?□��\�rd□�9H�F�□�����S\��$□�1�Ĭirk��□□l���HM�t�\�W�□*]�\□"�M..."
_id: ObjectId("61db79948de13a2a6519a679")

0: "�<8□□R□�Û�U□��rI"
1: "8�t�EG���$�      �□�ZI�□□��}��4/    □□□□V□□□□□y��□����V��N�h□�      �^�I{%l��H{:��..."
2: "j□18□□[□□□□f□□□KJa□□□/□ □UTc□rC□qGA@S□□□i□□);$×□□□�lh�d@□#�o□�I��6�□..."
3: "����>-��□*��"
_id: ObjectId("61db79948de13a2a6519a67a")

0: "i�4Uy;t�□VǴ:□□�hY□!8�"
1: "��'�□!R9"
_id: ObjectId("61db79948de13a2a6519a67b")

0: "���r!□���O���□□�"�N#�(��□Q<O-v�$˛��o|�bǵ$��PB�.�*ïG3□□;□□3□����03�6N□|□□..."
_id: ObjectId("61db79948de13a2a6519a67c")
```

There is a lot more documentation to be found on MongoDB recently, but this was not always a blessing. When I ran into this problem, I could only find solutions to these sort of problems, but related to the sample data from MongoDB tutorials. Or tutorials telling me how to import files via the Atlas browser Service. I found a stack that explained to me how to use the command line to import TSV-files.



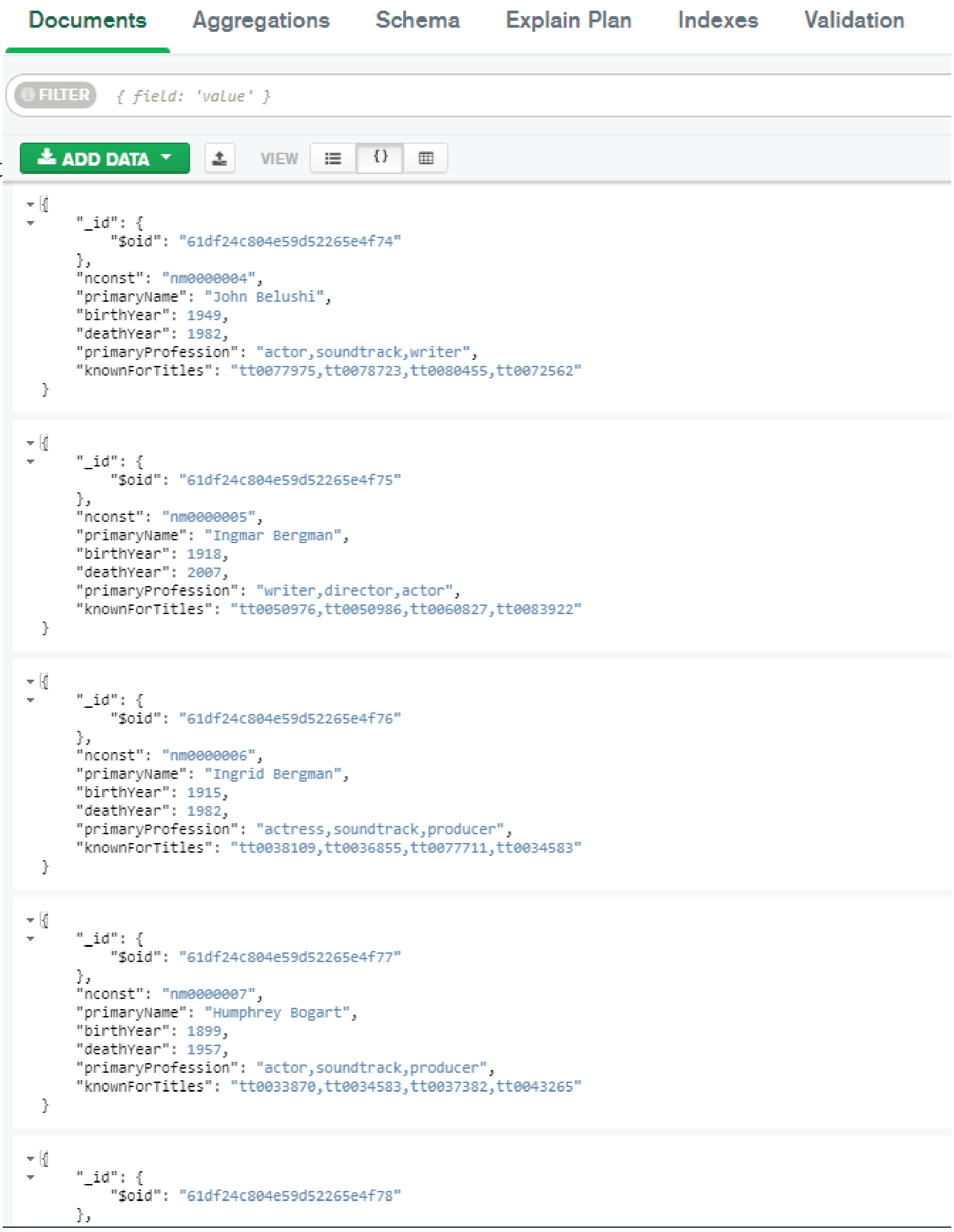
```
Minka@DESKTOP-U93G4M1 MINGW64 ~
$ mongoimport --db newImdb --collection newimdbcol --type tsv --file D://Downloads/name.basics.tsv/data.tsv --headerline
```

--db newImdb: the database I wanted to import into.
--collection newimdbcol: the collection I had already made in the UI.
--type tsv: the type has to be specified.
– file (etc): the path to the file on my computer.
– headerline: is necessary when using TSV or CSV files, otherwise mongoimport will import the first line as its own document.

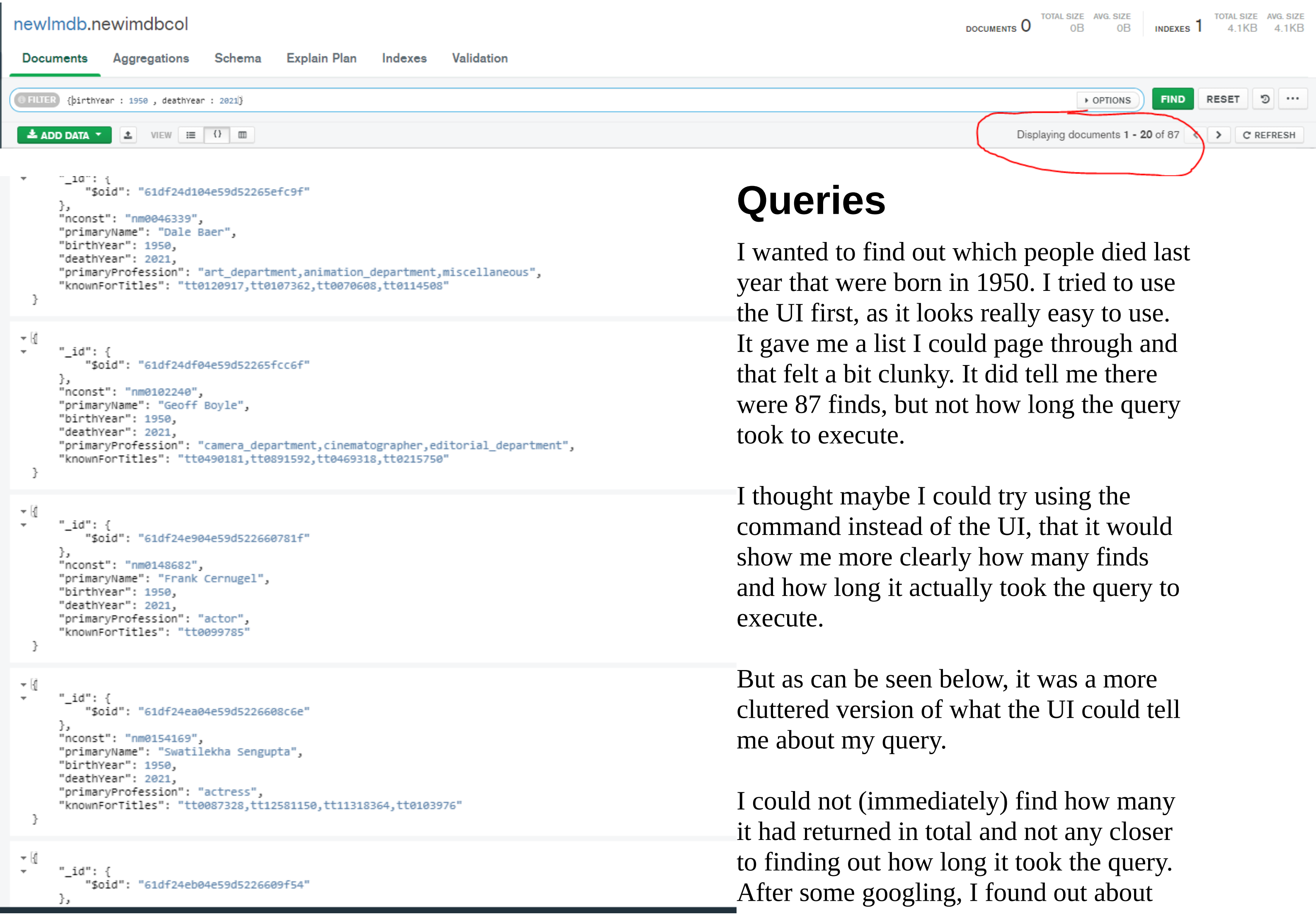
I had forgotten to download the MongoDB tools, so that was the first step. After adding the path, I was able to use the import command. It took about half an hour to actually import it, but it did work and I finally had something I could work with.

I tried some simple queries in the UI, to see what I could do with it, and decided after that what sort of query I wanted to use to do some simple testing.

The UI looks really easy to use. It has various tabs that allow you to add Indexes and aggregations. For some reason I totally looked over the Tab with *Explain Plan*. I kept searching for a way to find out how long each query took, and I thought I could only do it via the command line. Only afterwards I found a way to learn this using the UI.



```
{
    "_id": {
        "$oid": "61df24c804e59d52265e4f74"
    },
    "nconst": "nm0000004",
    "primaryName": "John Belushi",
    "birthYear": 1949,
    "deathYear": 1982,
    "primaryProfession": "actor,soundtrack,writer",
    "knownForTitles": "tt0077975,tt0078723,tt0080455,tt0072562"
}
{
    "_id": {
        "$oid": "61df24c804e59d52265e4f75"
    },
    "nconst": "nm0000005",
    "primaryName": "Ingmar Bergman",
    "birthYear": 1918,
    "deathYear": 2007,
    "primaryProfession": "writer,director,actor",
    "knownForTitles": "tt0050976,tt0050986,tt0060827,tt0083922"
}
{
    "_id": {
        "$oid": "61df24c804e59d52265e4f76"
    },
    "nconst": "nm0000006",
    "primaryName": "Ingrid Bergman",
    "birthYear": 1915,
    "deathYear": 1982,
    "primaryProfession": "actress,soundtrack,producer",
    "knownForTitles": "tt0038109,tt0036855,tt0077711,tt0034583"
}
{
    "_id": {
        "$oid": "61df24c804e59d52265e4f77"
    },
    "nconst": "nm0000007",
    "primaryName": "Humphrey Bogart",
    "birthYear": 1899,
    "deathYear": 1957,
    "primaryProfession": "actor,soundtrack,producer",
    "knownForTitles": "tt0033870,tt0034583,tt0037382,tt0043265"
}
{
    "_id": {
        "$oid": "61df24c804e59d52265e4f78"
    },
```



```
{birthYear : 1950 , deathYear : 2021}
```

```
    "_id": {
        "$oid": "61df24d104e59d52265efc9f"
    },
    "nconst": "nm0046339",
    "primaryName": "Dale Baer",
    "birthYear": 1950,
    "deathYear": 2021,
    "primaryProfession": "art_department,animation_department,miscellaneous",
    "knownForTitles": "tt0120917,tt0107362,tt0070608,tt0114508"
}

{
    "_id": {
        "$oid": "61df24df04e59d52265fcc6f"
    },
    "nconst": "nm0102240",
    "primaryName": "Geoff Boyle",
    "birthYear": 1950,
    "deathYear": 2021,
    "primaryProfession": "camera_department,cinematographer,editorial_department",
    "knownForTitles": "tt0490181,tt0891592,tt0469318,tt0215750"
}

{
    "_id": {
        "$oid": "61df24e904e59d522660781f"
    },
    "nconst": "nm0148682",
    "primaryName": "Frank Cernugel",
    "birthYear": 1950,
    "deathYear": 2021,
    "primaryProfession": "actor",
    "knownForTitles": "tt0099785"
}

{
    "_id": {
        "$oid": "61df24ea04e59d5226608c6e"
    },
    "nconst": "nm0154169",
    "primaryName": "Swatilekha Sengupta",
    "birthYear": 1950,
    "deathYear": 2021,
    "primaryProfession": "actress",
    "knownForTitles": "tt0087328,tt12581150,tt11318364,tt0103976"
}

{
    "_id": {
        "$oid": "61df24eb04e59d5226609f54"
    },
```

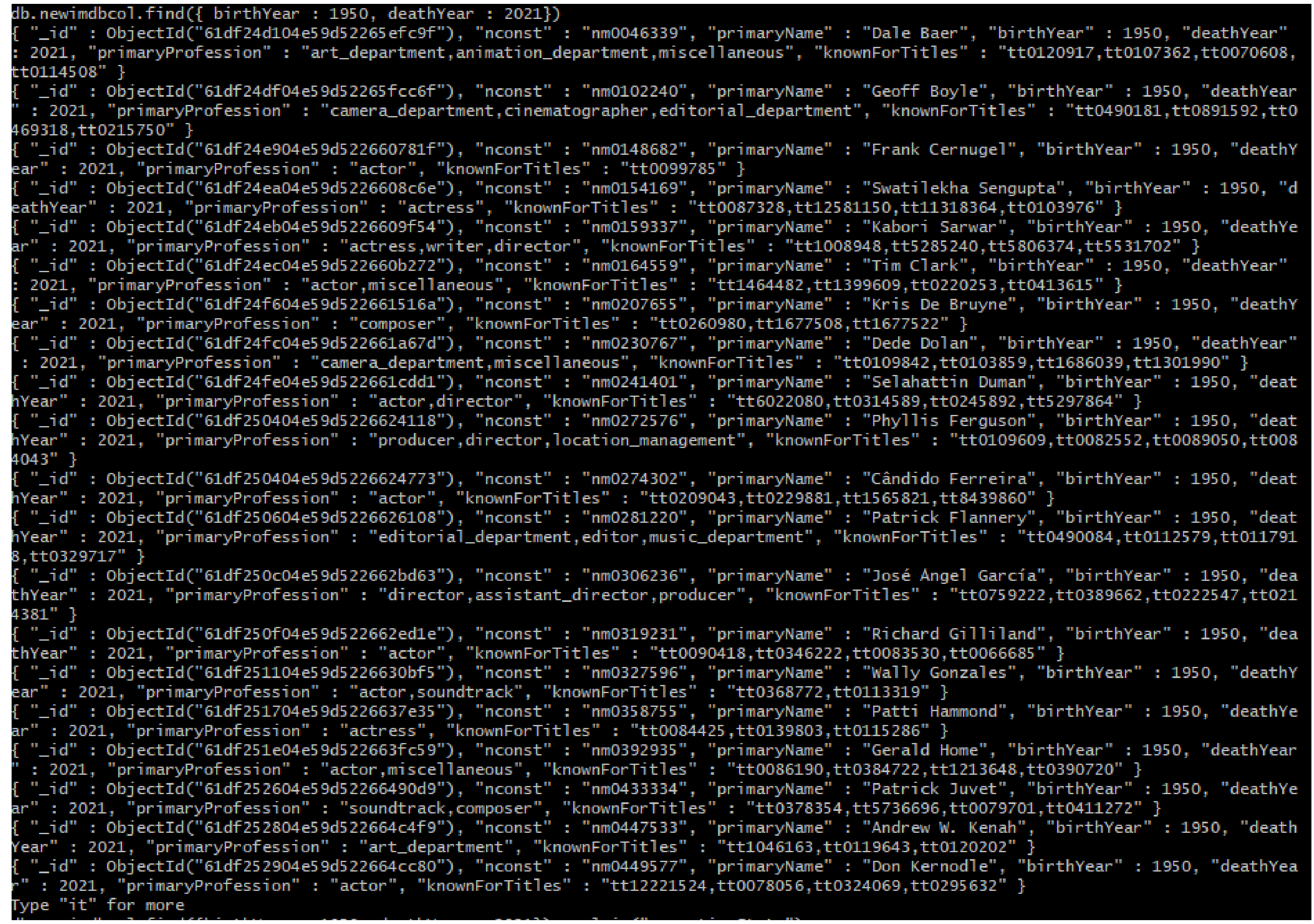
Queries

I wanted to find out which people died last year that were born in 1950. I tried to use the UI first, as it looks really easy to use. It gave me a list I could page through and that felt a bit clunky. It did tell me there were 87 finds, but not how long the query took to execute.

I thought maybe I could try using the command instead of the UI, that it would show me more clearly how many finds and how long it actually took the query to execute.

But as can be seen below, it was a more cluttered version of what the UI could tell me about my query.

I could not (immediately) find how many it had returned in total and not any closer to finding out how long it took the query. After some googling, I found out about



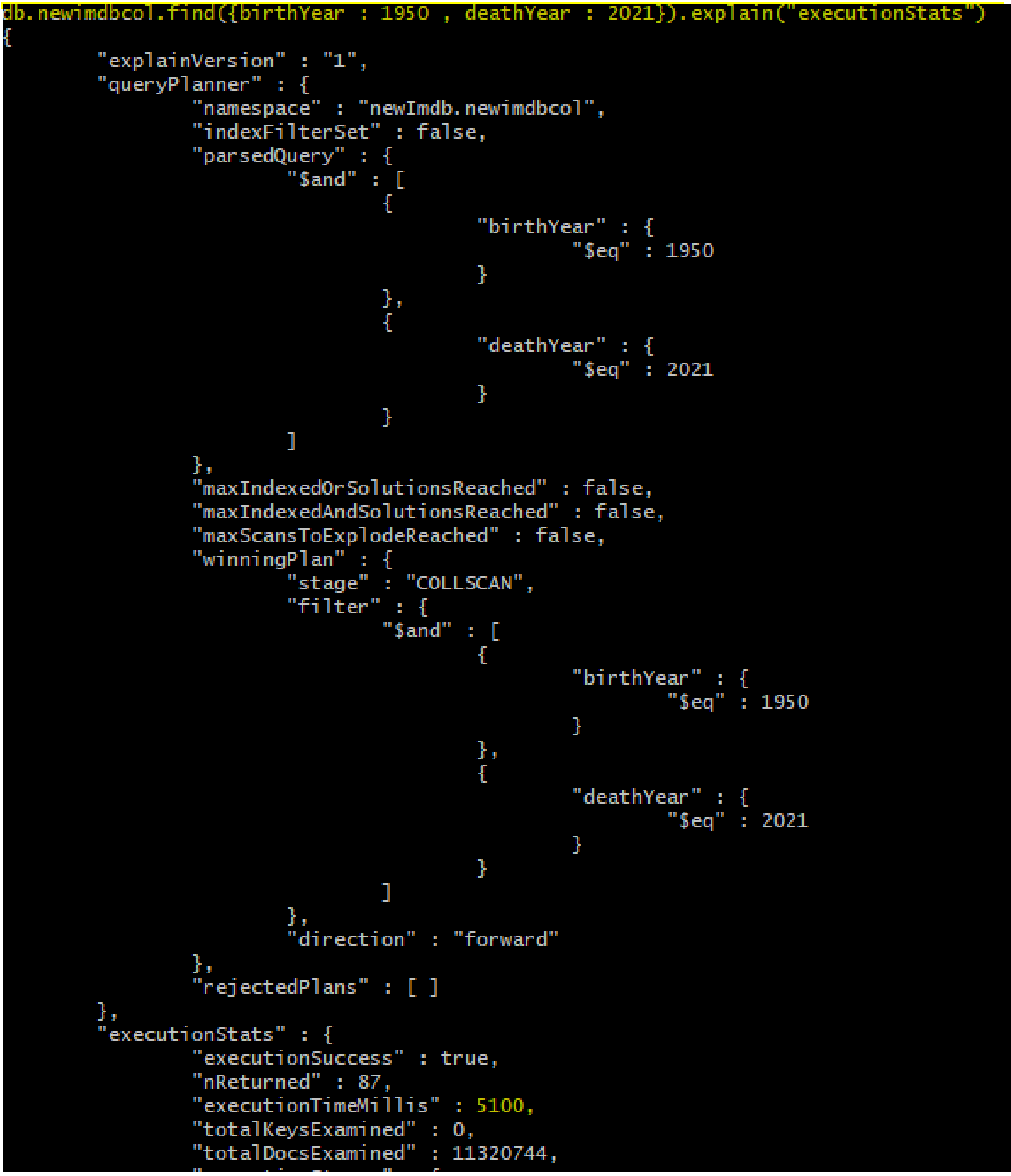
```
db.newimdbcol.find({ birthYear : 1950, deathYear : 2021})
{ "_id" : ObjectId("61df24d104e59d52265efc9f"), "nconst" : "nm0046339", "primaryName" : "Dale Baer", "birthYear" : 1950, "deathYear"
: 2021, "primaryProfession" : "art_department,animation_department,miscellaneous", "knownForTitles" : "tt0120917,tt0107362,tt0070608,
tt0114508" }
{ "_id" : ObjectId("61df24df04e59d52265fcc6f"), "nconst" : "nm0102240", "primaryName" : "Geoff Boyle", "birthYear" : 1950, "deathYear
" : 2021, "primaryProfession" : "camera_department,cinematographer,editorial_department", "knownForTitles" : "tt0490181,tt0891592,tt0
469318,tt0215750" }
{ "_id" : ObjectId("61df24e904e59d522660781f"), "nconst" : "nm0148682", "primaryName" : "Frank Cernugel", "birthYear" : 1950, "deathY
ear" : 2021, "primaryProfession" : "actor", "knownForTitles" : "tt0099785" }
{ "_id" : ObjectId("61df24ea04e59d5226608c6e"), "nconst" : "nm0154169", "primaryName" : "Swatilekha Sengupta", "birthYear" : 1950, "d
eathYear" : 2021, "primaryProfession" : "actress", "knownForTitles" : "tt0087328,tt12581150,tt11318364,tt0103976" }
{ "_id" : ObjectId("61df24eb04e59d5226609f54"), "nconst" : "nm0159337", "primaryName" : "Kabori Sarwar", "birthYear" : 1950, "deathYe
ar" : 2021, "primaryProfession" : "actress,writer,director", "knownForTitles" : "tt1008948,tt5285240,tt5806374,tt5531702" }
{ "_id" : ObjectId("61df24ec04e59d522660b272"), "nconst" : "nm0164559", "primaryName" : "Tim Clark", "birthYear" : 1950, "deathYear"
: 2021, "primaryProfession" : "actor,miscellaneous", "knownForTitles" : "tt1464482,tt1399609,tt0220253,tt0413615" }
{ "_id" : ObjectId("61df24f604e59d522661516a"), "nconst" : "nm0207655", "primaryName" : "Kris De Bruyne", "birthYear" : 1950, "deathY
ear" : 2021, "primaryProfession" : "composer", "knownForTitles" : "tt0260980,tt1677508,tt1677522" }
{ "_id" : ObjectId("61df24fc04e59d522661a67d"), "nconst" : "nm0230767", "primaryName" : "Dede Dolan", "birthYear" : 1950, "deathYear"
 : 2021, "primaryProfession" : "camera_department,miscellaneous", "knownForTitles" : "tt0109842,tt0103859,tt1686039,tt1301990" }
{ "_id" : ObjectId("61df24fe04e59d522661cdd1"), "nconst" : "nm0241401", "primaryName" : "Selahattin Duman", "birthYear" : 1950, "deat
hYear" : 2021, "primaryProfession" : "actor,director", "knownForTitles" : "tt6022080,tt0314589,tt0245892,tt5297864" }
{ "_id" : ObjectId("61df250404e59d5226624118"), "nconst" : "nm0272576", "primaryName" : "Phyllis Ferguson", "birthYear" : 1950, "deat
hYear" : 2021, "primaryProfession" : "producer,director,location_management", "knownForTitles" : "tt0109609,tt0082552,tt0089050,tt008
4043" }
{ "_id" : ObjectId("61df250404e59d5226624773"), "nconst" : "nm0274302", "primaryName" : "Cândido Ferreira", "birthYear" : 1950, "deat
hYear" : 2021, "primaryProfession" : "actor", "knownForTitles" : "tt0209043,tt0229881,tt1565821,tt8439860" }
{ "_id" : ObjectId("61df250604e59d5226626108"), "nconst" : "nm0281220", "primaryName" : "Patrick Flannery", "birthYear" : 1950, "deat
hYear" : 2021, "primaryProfession" : "editorial_department,editor,music_department", "knownForTitles" : "tt0490084,tt0112579,tt011791
8,tt0329717" }
{ "_id" : ObjectId("61df250c04e59d522662bd63"), "nconst" : "nm0306236", "primaryName" : "José Angel García", "birthYear" : 1950, "dea
thYear" : 2021, "primaryProfession" : "director,assistant_director,producer", "knownForTitles" : "tt0759222,tt0389662,tt0222547,tt021
4381" }
{ "_id" : ObjectId("61df250f04e59d522662ed1e"), "nconst" : "nm0319231", "primaryName" : "Richard Gilliland", "birthYear" : 1950, "dea
thYear" : 2021, "primaryProfession" : "actor", "knownForTitles" : "tt0090418,tt0346222,tt0083530,tt0066685" }
{ "_id" : ObjectId("61df251104e59d5226630bf5"), "nconst" : "nm0327596", "primaryName" : "Wally Gonzales", "birthYear" : 1950, "deathY
ear" : 2021, "primaryProfession" : "actor,soundtrack", "knownForTitles" : "tt0368772,tt0113319" }
{ "_id" : ObjectId("61df251704e59d522663 7e35"), "nconst" : "nm0358755", "primaryName" : "Patti Hammond", "birthYear" : 1950, "deathYe
ar" : 2021, "primaryProfession" : "actress", "knownForTitles" : "tt0084425,tt0139803,tt0115286" }
{ "_id" : ObjectId("61df251e04e59d522663fc59"), "nconst" : "nm0392935", "primaryName" : "Gerald Home", "birthYear" : 1950, "deathYear
" : 2021, "primaryProfession" : "actor,miscellaneous", "knownForTitles" : "tt0086190,tt0384722,tt1213648,tt0390720" }
{ "_id" : ObjectId("61df252604e59d52266490d9"), "nconst" : "nm0433334", "primaryName" : "Patrick Juvet", "birthYear" : 1950, "deathYe
ar" : 2021, "primaryProfession" : "soundtrack,composer", "knownForTitles" : "tt0378354,tt5736696,tt0079701,tt0411272" }
{ "_id" : ObjectId("61df252804e59d522664c4f9"), "nconst" : "nm0447533", "primaryName" : "Andrew W. Kenah", "birthYear" : 1950, "death
Year" : 2021, "primaryProfession" : "art_department", "knownForTitles" : "tt1046163,tt0119643,tt0120202" }
{ "_id" : ObjectId("61df252904e59d522664cc80"), "nconst" : "nm0449577", "primaryName" : "Don Kernodle", "birthYear" : 1950, "deathYea
r" : 2021, "primaryProfession" : "actor", "knownForTitles" : "tt12221524,tt0078056,tt0324069,tt0295632" }
Type "it" for more
```

the explain command, which showed me a lot of data *about* each query.

By adding *.explain("executionStats")* to my query, it showed me a lot of information about the query I tried.

I tried the query a few times with the explain command, to make sure I'd get a proper estimation of how long it took this query to execute. After a few tries, they were all between 5034 and 5220 milliseconds, so the screenshot on the right here, while not a real average, was close enough to it.

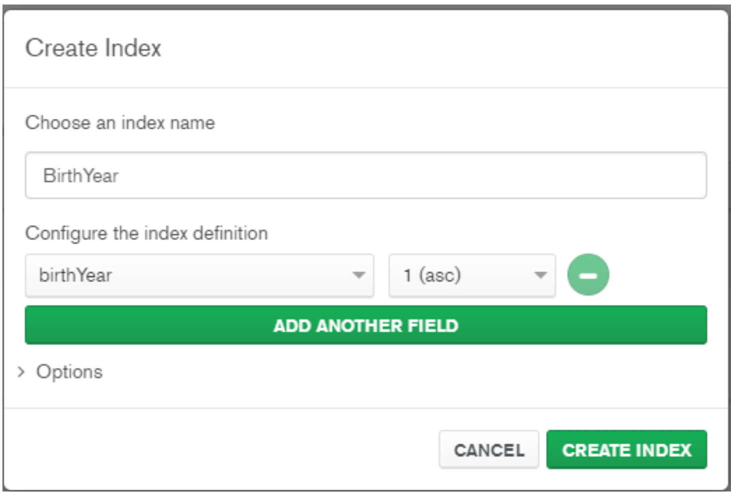
The UI also didn't tell me how many documents there actually were in the collection, but at the very bottom it tells us it went through more than eleven million documents and it took about 5100 milliseconds.



```
db.newimdbcol.find({birthYear : 1950 , deathYear : 2021}).explain("executionStats")
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "newImdb.newimdbcol",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "$and" : [
                                {
                                        "birthYear" : {
                                                "$eq" : 1950
                                        }
                                },
                                {
                                        "deathYear" : {
                                                "$eq" : 2021
                                        }
                                }
                        ]
                },
                "maxIndexedOrSolutionsReached" : false,
                "maxIndexedAndSolutionsReached" : false,
                "maxScansToExplodeReached" : false,
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "$and" : [
                                        {
                                                "birthYear" : {
                                                        "$eq" : 1950
                                                }
                                        },
                                        {
                                                "deathYear" : {
                                                        "$eq" : 2021
                                                }
                                        }
                                ]
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 87,
                "executionTimeMillis" : 5100,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 11320744,
```

Indexing

Since I was looking at birthYears firstly, I decided it would be wise to add an index for this field. I thought this way it wouldn't have to actually go through eleven millions documents. I decided to use the UI for this, as again, it looks really easy.
There is an option to add another field, so I was considering adding the deathYear as well, but I wanted to test each individually. I used an ascending index, because I was using 1950 as the birthYear.

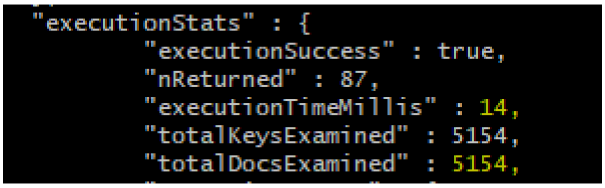


Create Index

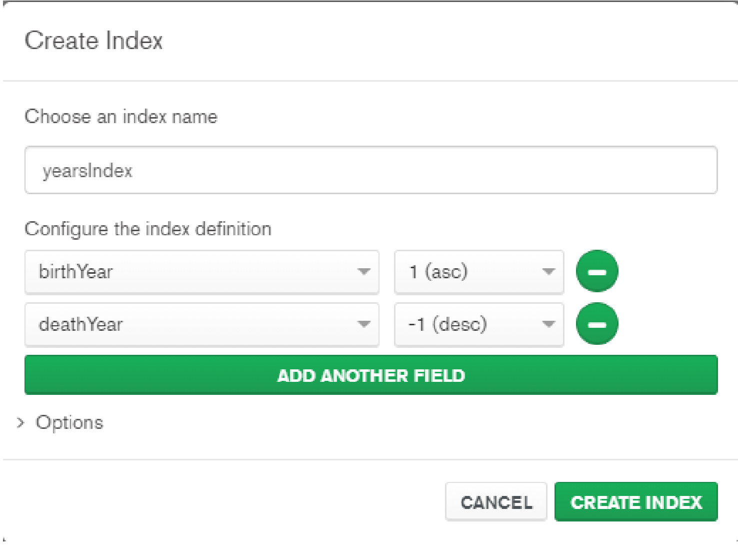
Choose an index name: BirthYear

Configure the index definition: birthYear | 1 (asc)

I immediately noticed the difference. Whereas before it took at least around 5 seconds before it would return the information, and this time it was almost immediate.



```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 87,
        "executionTimeMillis" : 14,
        "totalKeysExamined" : 5154,
        "totalDocsExamined" : 5154,
```



Create Index

Choose an index name

yearsIndex

Configure the index definition

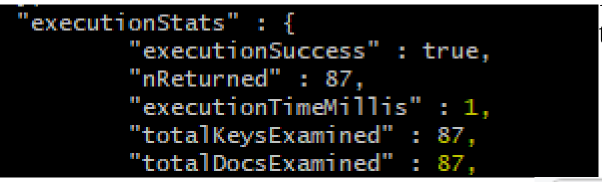
birthYear 1 (asc)

deathYear -1 (desc)

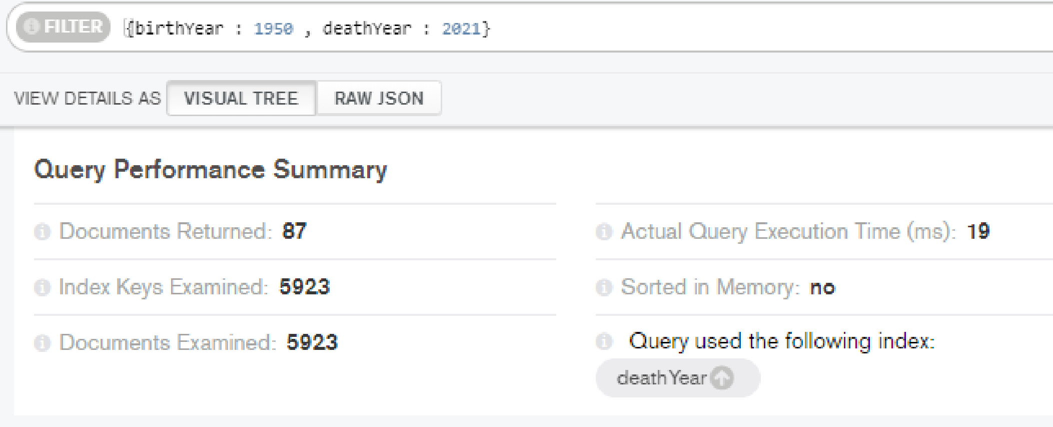
I was also correct in my estimation that it wouldn’t have to go through as many documents, as it only went through five thousand files this time. These are the documents of people that were born in 1950.

I decided to add another index for the deathYear this time, and I decided to make it a descending index, since I was using 2021 as the deathYear.

Unfortunately, while it returned basically immediately, the difference was no longer as obvious as after the first index (as was to be expected). It took only one millisecond to execute this time, and it only looked through 87 files.



```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 87,
        "executionTimeMillis" : 1,
        "totalKeysExamined" : 87,
        "totalDocsExamined" : 87,
```



FILTER {birthYear : 1950 , deathYear : 2021}

Query Performance Summary

Documents Returned: 87

Actual Query Execution Time (ms): 19

Index Keys Examined: 5923

Sorted in Memory: no

Documents Examined: 5923

Query used the following index:

deathYear

Only after doing these tests did I find a way of doing it in the UI. In the Explain plan, it tells us basically the same information as the shell would. (This screenshot is taken from a query with only the deathYear as index).

I think it is a little peculiar, because as I mentioned before, I ran every query a few times, to get a better feel for the estimation of the execution time. When I tried again with the index where I indexed both the birthYear and the deathYear, it would *always* tell me 0 milliseconds, while the command line would vary between either 1 and 0. I am putting this down to MongoDB somehow rounding up or down differently.

Indexing Findings

Index	Time to execute query[1] (in milliseconds)	Documents examined
No Index	5100	11.320.744
Only birthYear Index	14	5154
Only deathYear Index	19	5923
Both birth- & deathYear index	1	87

There is a slight difference between using a birthYear index or using a deathYear, because there are more people that died in 2021 than were born in 1950.

[1] These are based on the same query queried multiple times, and then choosing the option that was the closest to the estimated average.

Findings MongoDB

Installation

As mentioned, I didn’t really run into any problems with installing the proper tools. There were a lot of tutorials and documentation available.

Importing

Most tutorials want you to use the mongoDB atlas browser application. So that made importing a bit tricky at first, since the MongoCompass (The UI) only allows you to add JSON or CSV. I tried a lot of googling, but I only got results concerning how to import them via the Atlas Browser Application.

I think it is weird the UI allows *does* me to import the TSV document, but as an CSV document, but show all those weird symbols. It got me wondering if maybe the document was not compatible with the database, and was already looking into finding another dataset before I realised that I was simply using the wrong type of file.

Importing the data from the command line proved to be no problem at all.

Queries

It is a shame neither the UI or the Command Line shell allows you to just scroll through your query returns. In the UI you have to actually click through pages with twenty documents on each page, while the command line wanted you to *type “it” for more*. It would be nice to be able to just see everything in one go. I also hardly had any troubles with the language. It felt sort of intuitive, and while I had to google some of it at first, they came sort of naturally after a while.

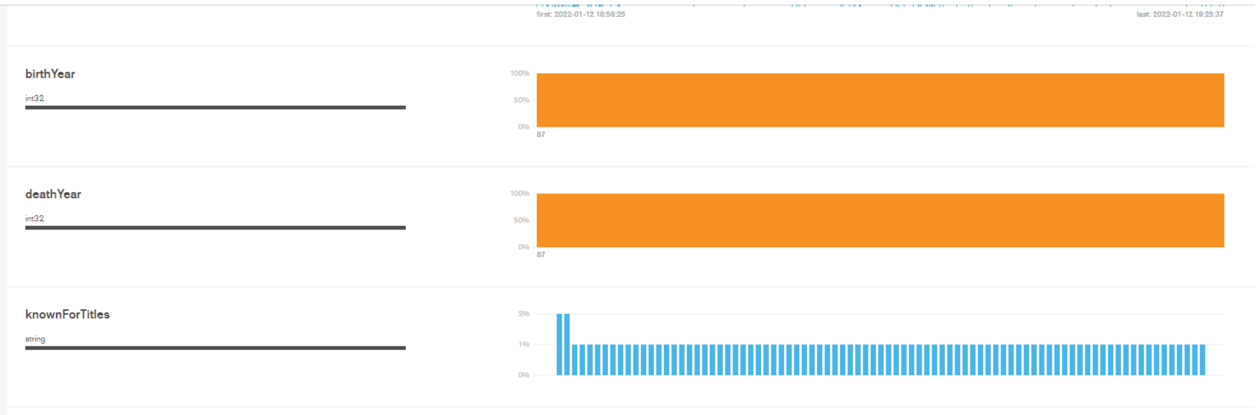
Indexing

I added indexes through the MongoDB Compass. It was very intuitive and easy to just add a new index. With just a few clicks I could add and delete the indexes to experiment.

I did try to do something with the aggregation stages in mongoDB, but I could not really figure it out, and with the amount of time left, decided to focus on indexing.

UI/command line

I really tried to use both, but I did notice a pattern in my way of working. I tended to make Databases and Collections in the UI, and then proceed to query from the command line. I also noted that even after I found out I could find ‘explanations’ of the queries in the UI, I still preferred to use the command line (mostly because I think it was slightly more accurate).



There was even a ‘schema’ tab that shows simple diagrams of the data you just received from the query. It wasn’t really interesting with this query, but I think it could be an easy way to analyse some data. It wouldn’t be really in-depth, but could provide some shallow analytical opportunities.

One of the best features of the MongoDB UI is that when you try to delete a database, collection or index etc. it doesn’t immediately delete it, but asks you to insert the name of whatever you are deleting at that moment. That was a good safeguard from accidentally clicking the delete button and dropping everything.

However sometimes after I would drop a table or collection, and add a new one with the same name, the UI would respond saying that collection or database name would already exist.

Pros and Cons MongoDB

Pros	**Cons**
Beginner Friendly	Only JSON or CSV files option available in the UI
Mostly clear UI	Joining Documents can be tricky*
Lot of documentation available	Lot of documentation in a certain direction available
Easy to use Querying language	Not easy to deal with multi-document transactions*
Quick query performance	Inbuilt aggregation tab was a bit confusing.
Easily Scalable*	
Free to use	
Delete safeguard	

* These are aspects that I haven’t experimented with myself, but I read about on the internet. There were a lot of other pros and cons listed, but after doing the research for the theory this module and experimenting with MongoDB, I could see where these were coming from.

Bruce, D. (2021, 21 april). *Understanding the Pros and Cons of MongoDB*. Knowledgenile. Geraadpleegd op 13 januari 2022, van https://www.knowledgenile.com/blogs/pros-and-cons-of-mongodb/

Singh, A. (z.d.). *Mongodb advantages and disadvantages*. Interview Questions Angular, JavaScript, Java, PHP, SQL, C#, Vue, NodeJs, ReactJs. Geraadpleegd op 13 januari 2022, van https://www.code-sample.com/2016/07/mongodb-advantages-disadvantages.html

TrustRadius CAPTCHA. (z.d.). TrustRadius. Geraadpleegd op 13 januari 2022, van https://www.trustradius.com/products/mongodb/reviews?qs=pros-and-cons#reviews