Домашняя работа №1. MongoDB

Автор: Васильев Дмитрий Олегович

Цель:

В результате выполнения ДЗ научится разворачивать MongoDB, заполнять данными и делать запросы.

Описание/инструкция выполнения домашнего задания:

- 1. Установить MongoDB одним из способов: локально, докер или ВМ(облачный сервис);
- 2. Заполнить данными (примеры датасетов https://habr.com/ru/company/edison/blog/480408/);
- 3. Написать несколько запросов на выборку и обновление данных;
- 4. Создать индексы и сравнить производительность.

In [2]:

```
from pymongo import MongoClient

client = MongoClient('mongodb://root:example@localhost:27017/')
db_name = 'lesson'
collection_name = 'customers'
```

1. Установил через Docker Compose

In [7]:

```
! cat ../docker-compose.yml
version: "3"
services:
 mongo:
    image: mongo
    restart: always
      - "27017:27017"
    environment:
      MONGO INITDB ROOT USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: example
 mongo-express:
    image: mongo-express
    restart: always
    ports:
      - 8081:8081
    environment:
      ME CONFIG MONGODB ADMINUSERNAME: root
      ME CONFIG MONGODB ADMINPASSWORD: example
      ME CONFIG MONGODB URL: mongodb://root:example@mongo:27017/
```

2. Заполнил данными из <u>следующего датасета</u> (<u>https://www.kaggle.com/datasets/shwetabh123/mall-customers</u>).

In [1]:

```
! head -n 10 Mall_Customers.csv
from pathlib import Path

dataset_path = Path('Mall_Customers.csv')
```

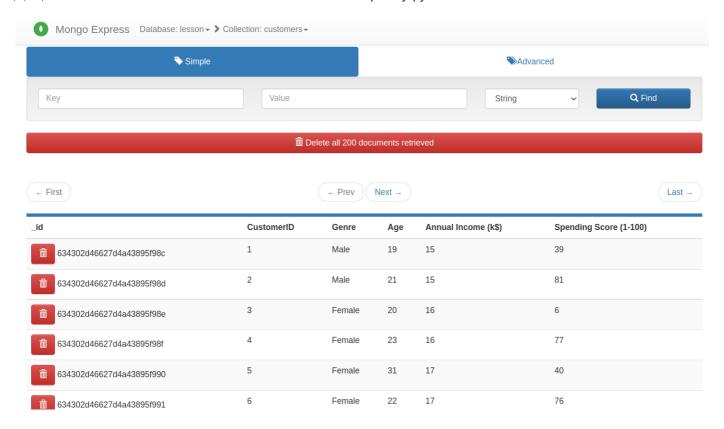
```
CustomerID, Genre, Age, Annual Income (k$), Spending Score (1-100) 0001, Male, 19, 15, 39 0002, Male, 21, 15, 81 0003, Female, 20, 16, 6 0004, Female, 23, 16, 77 0005, Female, 31, 17, 40 0006, Female, 22, 17, 76 0007, Female, 35, 18, 6 0008, Female, 23, 18, 94 0009, Male, 64, 19, 3
```

Импорт данных:

In [3]:

```
import pandas as pd
from pymongo import MongoClient
import json

def mongo_import_csv(csv_path, db_name, collection_name, client) -> int:
    db = client[db_name]
    collection = db[collection_name]
    data = pd.read_csv(csv_path)
    payload = json.loads(data.to_json(orient='records'))
    collection.delete_many({})
    collection.insert_many(payload)
    return collection
collection = mongo_import_csv(dataset_path, db_name, collection_name, client)
```



3. Напишем несколько запросов на выборку и обновление данных:

In [9]:

```
data = collection.find({'Age': {'$gt': 18}})
list(data)
Out[9]:
[{' id': ObjectId('634307b9994f5f97a91e4b29'),
  'CustomerID': 34,
  'Genre': 'Male',
  'Age': 18,
  'Annual Income (k$)': 33,
  'Spending Score (1-100)': 92},
 {'_id': ObjectId('634307b9994f5f97a91e4b49'),
  CustomerID': 66,
  'Genre': 'Male',
  'Age': 18,
  'Annual Income (k$)': 48,
  'Spending Score (1-100)': 59},
 {' id': ObjectId('634307b9994f5f97a91e4b63'),
  'CustomerID': 92,
  'Genre': 'Male',
  'Age': 18,
  'Annual Income (k$)': 59,
  'Spending Score (1-100)': 41},
 {' id': ObjectId('634307b9994f5f97a91e4b7a'),
  'CustomerID': 115,
  'Genre': 'Female',
  'Age': 18,
  'Annual Income (k$)': 65,
  'Spending Score (1-100)': 48}]
```

```
In [23]:
```

```
res = collection.update_one({'CustomerID': 34}, {"$set": { 'Genre': 'Female' }})
%timeit data = collection.find({'CustomerID': {'$eq': 34}})
list(data)

6.25 μs ± 826 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops e ach)
```

Out[23]:

[]

4. Создать индексы и сравнить производительность.

In [17]:

```
from pymongo import ASCENDING
collection.create_index([('CustomerID', ASCENDING)])
```

Out[17]:

'CustomerID 1'

In [24]:

```
res = collection.update_one({'CustomerID': 34}, {"$set": { 'Genre': 'Male' }})
%timeit data = collection.find({'CustomerID': {'$eq': 34}})
list(data)
```

```
5.5~\mu s~\pm~51.3~ns per loop (mean \pm~std. dev. of 7 runs, 100,000 loops e ach)
```

Out[24]:

[]

Вывод:

В 1-ом случае результат $6.25~\mu s~\pm~826~ns~per~loop$ (до создания индекса по полю CustomerID), во 2-ом $5.5~\mu s~\pm~51.3~ns~per~loop$.