Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по Лабораторной Работе № 5

по дисциплине «**Проектирование и реализация баз данных**»

Mongo DB

Автор: Мазеин Никита Олегович

Факультет: ФИКТ

Группа: K32402

Преподаватель: Говорова Марина Михайловна

Санкт-Петербург

2023

1. Цель работы:

   овладеть практическими навыками работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.

2. Выполнение:

ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ

Задание 8.1.1.

1) Создайте базу данных learn

```
> db.unicorns.countDocuments()
< 0
learn>
```

2) Заполните коллекцию единорогов unicorns:

```
  db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
  db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649546641b278a378a14cf3c")
    }
  }
> db.unicorns.countDocuments()
< 11
learn>
```

3) Используя второй способ, вставьте в коллекцию единорогов документ:

```
> doc = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
  )
< {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
> db.unicorns.insert(doc)
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649546c21b278a378a14cf3d")
    }
  }
> db.unicorns.countDocuments()
< 12
learn>
```

4) Проверьте содержимое коллекции с помощью метода find

ВЫБОРКА ДАННЫХ ИЗ БД

Задание 8.1.2.

1) Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.

```
> db.unicorns.find({gender:'m'}).sort({name : 1}).limit(3)
< {
    _id: ObjectId("649546c21b278a378a14cf3d"),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId("649546641b278a378a14cf32"),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
```

```
> db.unicorns.find({gender:'f'}).sort({name : 1}).limit(3)
< {
    _id: ObjectId("649546641b278a378a14cf33"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId("649546641b278a378a14cf37"),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
```

2) Найдите всех самок, которые любят carrot. Ограничьте этот список первой особью с помощью функций findOne и limit.

```
> db.unicorns.findOne({gender:'f', loves:'carrot'})
< {
    _id: ObjectId("649546641b278a378a14cf33"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
> db.unicorns.find({gender:'f', loves:'carrot'}).limit(1)
< {
    _id: ObjectId("649546641b278a378a14cf33"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
```

Задание 8.1.3.

1) Модифицируйте запрос для вывода списков самцов единорогов, исключив из результата информацию о предпочтениях и поле

```
> db.unicorns.find({ gender: 'm' }, { loves: 0, gender: 0, _id: 0 }).sort({ $natural: -1 }).limit(3)
< {
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
  {
    name: 'Pilot',
    weight: 650,
    vampires: 54
  }
  {
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  }
```

Задание 8.1.4.

1) Вывести список единорогов в обратном порядке добавления

```
> db.unicorns.find().sort({ $natural: -1 })
< {
    _id: ObjectId("649546c21b278a378a14cf3d"),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId("649546641b278a378a14cf3c"),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

Задание 8.1.5.

1) Вывести список единорогов с названием первого любимого предпочтения, исключив идентификатор

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Задание 8.1.6.

1) Вывести список самок единорогов весом от полтонны до 700 кг, исключив вывод идентификатора

```
> db.unicorns.find({weight: {$gte:500, $lt:700}}, {_id:0})
< {
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    name: 'Roooooodles',
    loves: [
      'apple'
    ],
    weight: 575,
    gender: 'm',
    vampires: 99
  }
```

Задание 8.1.7.

1) Вывести список самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора

```
> db.unicorns.find({weight: {$gte:500, $lt:700}, loves: {$all:['grape','lemon']}}, {_id:0})
< {
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
```

Задание 8.1.8.

1) Найти всех единорогов, не имеющих ключ vampires

```
> db.unicorns.find({vampires:{$exists:false}}, {_id:0})
< {
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

Задание 8.1.9.

1) Вывести список упорядоченный список имен самцов единорогов с информацией об их первом предпочтении

```
> db.unicorns.find({gender:'m'}, {loves: { $slice: 1}, _id:0})
< {
    name: 'Horny',
    loves: [
      'carrot'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    name: 'Unicrom',
    loves: [
      'energon'
    ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
```

ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB.

ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ

ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ

Задание 8.2.1.

1) Создайте коллекцию towns, включающую следующие документы:

```
> db.createCollection('towns')
< { ok: 1 }
> doc1 = {name: "Punxsutawney ",
  populatiuon: 6200,
  last_sensus: ISODate("2008-01-31"),
  famous_for: [""],
  mayor: {
     name: "Jim Wehrle"
  }}
< {
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: 2008-01-31T00:00:00.000Z,
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  }
```

```
> doc2 = {name: "New York",
  populatiuon: 22200000,
  last_sensus: ISODate("2009-07-31"),
  famous_for: ["status of liberty", "food"],
  mayor: {
     name: "Michael Bloomberg",
  party: "I"}}
< {
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
> doc3 = {name: "Portland",
  populatiuon: 528000,
  last_sensus: ISODate("2009-07-20"),
  famous_for: ["beer", "food"],
  mayor: {
     name: "Sam Adams",
  party: "D"}}
```

```
> db.towns.insertMany([doc1,doc2,doc3])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("64955ccd1b278a378a14cf3e"),
      '1': ObjectId("64955ccd1b278a378a14cf3f"),
      '2': ObjectId("64955ccd1b278a378a14cf40")
    }
  }
> db.towns.countDocuments()
< 3
```

2) Сформировать запрос, который возвращает список городов с независимыми мэрами (party="I"). Вывести только название города и информацию о мэре

```
> db.towns.find({"mayor.party":"I"}, {name:1,mayor:1,_id:0})
< {
    name: 'New York',
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
```

3) Сформировать запрос, который возвращает список беспартийных мэров (party отсутствует). Вывести только название города и информацию о мэре.

```
> db.towns.find({"mayor.party":{$exists:false}}, {name:1,mayor:1,_id:0})
< {
    name: 'Punxsutawney ',
    mayor: {
      name: 'Jim Wehrle'
    }
  }
```

Задание 8.2.2.

1) Сформировать функцию для вывода списка самцов единорогов

```
learn> function printMaleUnicorns(){
         const maleUnicorns = db.unicorns.find({gender:'m'}, {_id:0});
         maleUnicorns.sort({name:1}).forEach((unicorn)=>{
           print('Name:', unicorn.name);
           print('Preferences:', unicorn.loves)
           print('Weight:', unicorn.weight);
           print('Vampire:', unicorn.vampires);
           print('.............................');
         });
       }
```

```
> printMaleUnicorns()
< Name:
< Dunx
< Preferences:
< [ 'grape', 'watermelon' ]
< Weight:
< 704
< Vampire:
< 165
< .............................
< Name:
< Horny
< Preferences:
< [ 'carrot', 'papaya' ]
< Weight:
< 600
< Vampire:
< 63
< .............................
< Name:
< Kenny
< Preferences:
< [ 'grape', 'lemon' ]
```

2) Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке

```
learn> unicornCursor = db.unicorns.find().sort({ name: 1 }).limit(2); null;
```

```
learn> unicornCursor.forEach(unicorn => {
         print('Name:', unicorn.name);
         print('Preferences:', unicorn.loves);
         print('Weight:', unicorn.weight);
         print('Vampires:', unicorn.vampires);
         print('-------------');
       });
```

```
> unicornCursor = db.unicorns.find().sort({ name: 1 }).limit(2); null;
< null
> unicornCursor.forEach(unicorn => {
    print('Name:', unicorn.name);
    print('Preferences:', unicorn.loves);
    print('Weight:', unicorn.weight);
    print('Vampires:', unicorn.vampires);
    print('-------------');
  });
< Name:
< Aurora
< Preferences:
< [ 'carrot', 'grape' ]
< Weight:
< 450
< Vampires:
< 43
< -------------
< Name:
< Ayna
< Preferences:
< [ 'strawberry', 'lemon' ]
< Weight:
```

Задание 8.2.3.

1) Вывести количество самок единорогов весом от полутонны до 600 кг

```
> db.unicorns.find({gender:'f',weight: {$gte:500, $lt:600}}).count()
< 2
```

Задание 8.2.4.

1) Вывести список предпочтений

```
> db.unicorns.distinct("loves")
< [
    'apple',      'carrot',
    'chocolate',  'energon',
    'grape',      'lemon',
    'papaya',     'redbull',
    'strawberry', 'sugar',
    'watermelon'
  ]
```

Задание 8.2.5.

1) Посчитать количество особей единорогов обоих полов

```
> db.unicorns.aggregate({"$group":{_id:"$gender",count:{$sum:1}}})
< {
    _id: 'm',
    count: 7
  }
  {
    _id: 'f',
    count: 5
  }
```

Задание 8.2.6.

1) Выполнить команду:

```
> db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
✖ ▸ TypeError: db.unicorns.save is not a function
```

NOTE

Starting in MongoDB 4.2, the `db.collection.save()` method is deprecated. Use `db.collection.insertOne()` or `db.collection.replaceOne()` instead.

```
> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
< {
    acknowledged: true,
    insertedId: ObjectId("64956e441b278a378a14cf41")
  }
```

2) Проверить содержимое коллекции unicorns

```
{
  _id: ObjectId("64956e441b278a378a14cf41"),
  name: 'Barny',
  loves: [
    'grape'
  ],
  weight: 340,
  gender: 'm'
}
```

Задание 8.2.7.

1) Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вапмира

```
> db.unicorns.updateOne({name:"Ayna"}, {$set: {weight:800, vampires:51}},{upsert:true})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

2) Проверить содержимое коллекции unicorns

```
> db.unicorns.find({name:"Ayna"})
< {
    _id: ObjectId("649546641b278a378a14cf37"),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
```

Задание 8.2.8.

1) Для самца единорога Raleigh внести изменения в БД: теперь он любит рэдбул
2) Проверить содержимое коллекции unicorns

```
> db.unicorns.updateOne({name:"Raleigh"}, {$set: {loves:["redbull"]}},{upsert:true})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name:"Raleigh"})
< {
    _id: ObjectId("649546641b278a378a14cf39"),
    name: 'Raleigh',
    loves: [
      'redbull'
    ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
```

Задание 8.2.9.

1) Всем самцам единорогов увеличить количество убитых вапмиров на 5

```
> db.unicorns.update({gender:"m"},{$inc:{vampires:5}})
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

2) Проверить содержимое коллекции unicorns

```
> db.unicorns.find({gender:"m"})
< {
    _id: ObjectId("649546641b278a378a14cf32"),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 68
  }
  {
    _id: ObjectId("649546641b278a378a14cf34"),
    name: 'Unicrom',
    loves: [
      'energon',
      'redbull'
    ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
```

Задание 8.2.10.

1) Изменить информацию о городе Портланд: мэр этого города теперь беспартийный

```
{
  _id: ObjectId("64955ccd1b278a378a14cf40"),
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams',
    party: 'D'
  }
```

```
> db.towns.update({name:"Portland"}, {$unset: {"mayor.party":1}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

2) Проверить содержимое коллекции towns

```
> db.towns.find({name:"Portland"})
< {
    _id: ObjectId("64955ccd1b278a378a14cf40"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [
      'beer',
      'food'
    ],
    mayor: {
      name: 'Sam Adams'
    }
```

Задание 8.2.11.

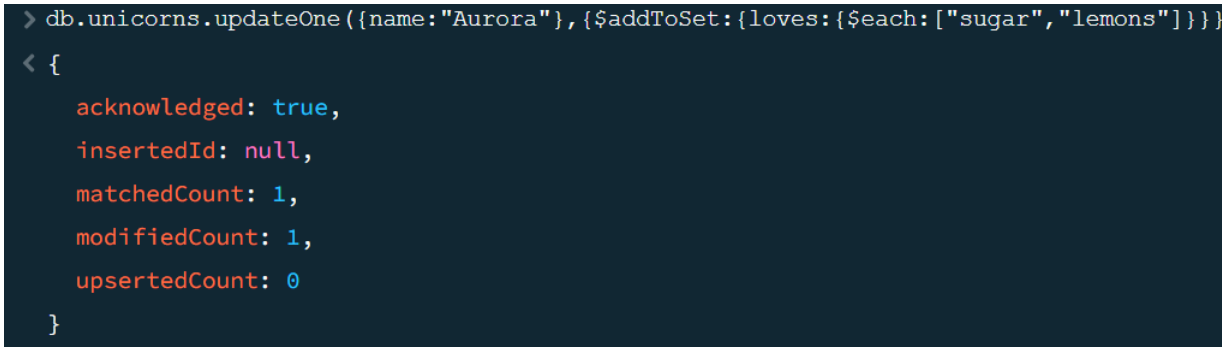
1) Изменить информацию о самце единорога Pilot: теперь он любит и шоколад

```
> db.unicorns.updateOne({name:"Pilot"},{$push:{loves:"chocolate"}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
```

2) Проверить содержимое коллекции unicorns

Задание 8.2.12.

1) Изменить информацию о самке единорога Aurora: теперь она любит еще и сахар, и лимоны



2) Проверить содержимое коллекции unicorns

```
> db.unicorns.find({name:"Aurora"})
< {
    _id: ObjectId("649546641b278a378a14cf33"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar',
      'lemons'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

УДАЛЕНИЕ ДАННЫХ ИЗ КОЛЛЕКЦИИ

Задание 8.2.13.

1) Создайте коллекцию towns, включающую следующие документы:

```
> db.towns.insertMany([doc1,doc2,doc3])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649588461b278a378a14cf42"),
      '1': ObjectId("649588461b278a378a14cf43"),
      '2': ObjectId("649588461b278a378a14cf44")
    }
```

2) Удалите документы с беспартийными мэрами

```
> db.towns.remove({"mayor.party":{$exists:0}})
< {
    acknowledged: true,
    deletedCount: 1
  }
```

3) Проверьте содержание коллекции

```
> db.towns.countDocuments()
< 2
```

4) Очистите коллекцию

```
> db.towns.remove({})
< {
    acknowledged: true,
    deletedCount: 2
  }
```

5) Просмотрите список доступных коллекций

```
> db.getCollectionNames()
< [ 'learn', 'unicorns', 'towns' ]
```

ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

ССЫЛКИ В БД

Задание 8.3.1.

1) Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание

```
learn > db.habitats.insertMany([
        {
          "_id": "rom",
          "name": "Romania",
          "description": "Welcome to Romania, the land of sparkling vampires and enchanting castles. Watch out for Dracula's tea parties!"
        },
        {
          "_id": "fra",
          "name": "France",
          "description": "Bienvenue to France, where the Eiffel Tower doubles as a croissant holder and berets are required to enter the Louvre."
        },
        {
        "_id": "aus",
        "name": "Australia",
        "description": "G'day mate! Welcome to Australia, where kangaroos hop around with their pet koalas and the beaches are guarded by surfing crocodiles
        }
      ]);
```

```
< {
    acknowledged: true,
    insertedIds: {
      '0': 'rom',
      '1': 'fra',
      '2': 'aus'
    }
  }
```

2) Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания

```
> db.unicorns.update({_id:ObjectId("649546641b278a378a14cf32")},{$set : {habitat:{$ref:"habitat", $id: "rom"}}
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

3) Проверьте содержание коллекции едиорогов

```
> db.unicorns.find({_id:ObjectId("649546641b278a378a14cf32")})
< {
    _id: ObjectId("649546641b278a378a14cf32"),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    habitat: DBRef("habitat", 'rom')
```

НАСТРОЙКА ИНДЕКСОВ

Задание 8.3.2.

1) Проверьте, можно ли задать для коллекции unicorns индекс для ключа name с флагом unique

```
> db.unicorns.ensureIndex({"name" : 1}, {"unique" : true})
< [ 'name_1' ]
> db.unicorns.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
  ]
```

УПРАВЛЕНИЕ ИНДЕКСАМИ

Задание 8.3.3.

1) Получите информацию обо всех индексах коллекции unicorns

```
> db.unicorns.ensureIndex({"name" : 1}, {"unique" : true})
< [ 'name_1' ]
> db.unicorns.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
  ]
```

2) Удалите все индексы, кроме индекса для идентификатора

```
> db.unicorns.dropIndex({name: 1})
< { nIndexesWas: 2, ok: 1 }
> db.unicorns.getIndexes()
< [ { v: 2, key: { _id: 1 }, name: '_id_' } ]
```

3) Попытайтесь удалить индекс для идентификатора

```
> db.unicorns.dropIndex({_id: 1})
⊗ ▸ MongoServerError: cannot drop _id index
```

ПЛАН ЗАПРОСА

Задание 8.3.4.

1) Создайте объемную коллекцию numbers, задействовав курсор:

```
> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649596121b278a378a1655e4")
    }
  }
```

2) Выберите последних четыре документа

```
> db.numbers.find().sort({value: -1}).limit(4)
< {
    _id: ObjectId("649596121b278a378a1655e4"),
    value: 99999
  }
  {
    _id: ObjectId("649596121b278a378a1655e3"),
    value: 99998
  }
  {
    _id: ObjectId("649596121b278a378a1655e2"),
    value: 99997
  }
  {
    _id: ObjectId("649596121b278a378a1655e1"),
    value: 99996
  }
```

3) Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра executionTimeMillis)

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 57,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
  executionStages: {
    stage: 'SORT',
    nReturned: 4,
    executionTimeMillisEstimate: 0,
```

4) Создайте индекс для ключа value

```
> db.numbers.createIndex({value:1})
< value_1
```

5) Получите информацию о всех индексах коллекции numbers

```
> db.numbers.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { value: 1 }, name: 'value_1' }
  ]
```

6) Выполните запрос 2. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса? `executionTimeMillis: 44,`
7) Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен? 57 -44 = 13ms boost Эффективнее второй запрос с индексацией.

Выводы:

В данной лабораторной работе было выполнено ознакомление с MongoDB, его командами. Была осуществлена работа с несколькими коллекциями базы данных, совершалось добавление элементов, их обновление и удаление, работа с ограничением вывода. Была создана коллекция, на которую совершались ссылки с другой коллекции. Осуществлено добавление, настройка и удаление индексов, анализ поиска по индексу и без него.