Министерство науки и высшего образования Российской Федерации ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

(Университет ИТМО)

Факультет прикладной информатики

Образовательная программа Мобильные и сетевые технологии

Направление подготовки 09.03.03 Мобильные и сетевые технологии

О Т Ч Е Т

Лабораторная работа 6

Тема задания: «Работа с БД в СУБД MongoDB»

Обучающийся: Анисимов Владислав Андреевич, группа К3240

Проверяющий: Говорова М.М., преподаватель

Санкт−Петербург,

2025

СОДЕРЖАНИЕ

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.

Практическое задание

Привести результаты выполнения практических заданий (номер задания, формулировка, команда, лог (скриншот) результата, вывод (при необходимости)).

Выполнение

1 Лабораторная 6.1

1) Проверьте работоспособность системы запуском клиента mongo.

```
Current Mongosh Log ID: 68360a5d24c661d3cfc73bf7
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.3.1
Using MongoDB:          8.0.9
Using Mongosh:          2.3.1

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

------
   The server generated these startup warnings when booting
   2025-05-27T21:12:18.182+03:00: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------
```

2) Выполните методы

a) db.help()

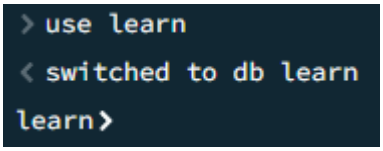
```
> db.help
< Database Class

  getMongo                  Returns the current database connection
  getName                   Returns the name of the DB
  getCollectionNames        Returns an array containing the names of all collections in the current database.
  getCollectionInfos        Returns an array of documents with collection information, i.e. collection name and options, for the current database.
  runCommand                Runs an arbitrary command on the database.
  adminCommand              Runs an arbitrary command against the admin database.
  aggregate                 Runs a specified admin/diagnostic pipeline which does not require an underlying collection.
  getSiblingDB              Returns another database without modifying the db variable in the shell environment.
  getCollection             Returns a collection or a view object that is functionally equivalent to using the db.<collectionName>.
  dropDatabase              Removes the current database, deleting the associated data files.
  createUser                Creates a new user for the database on which the method is run. db.createUser() returns a duplicate user error if the user already exists on the
                            database.
  updateUser                Updates the user's profile on the database on which you run the method. An update to a field completely replaces the previous field's values. This
                            includes updates to the user's roles array.
```

b) db.help

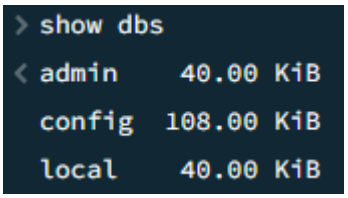
```
> db.help()
< Database Class

  getMongo                  Returns the current database connection
  getName                   Returns the name of the DB
  getCollectionNames        Returns an array containing the names of all collections in the current database.
  getCollectionInfos        Returns an array of documents with collection information, i.e. collection name and options, for the current database.
  runCommand                Runs an arbitrary command on the database.
  adminCommand              Runs an arbitrary command against the admin database.
  aggregate                 Runs a specified admin/diagnostic pipeline which does not require an underlying collection.
  getSiblingDB              Returns another database without modifying the db variable in the shell environment.
  getCollection             Returns a collection or a view object that is functionally equivalent to using the db.<collectionName>.
  dropDatabase              Removes the current database, deleting the associated data files.
  createUser                Creates a new user for the database on which the method is run. db.createUser() returns a duplicate user error if the user already exists on the
                            database.
  updateUser                Updates the user's profile on the database on which you run the method. An update to a field completely replaces the previous field's values. This
                            includes updates to the user's roles array.
```

c) db.stats()

3) Создайте БД learn

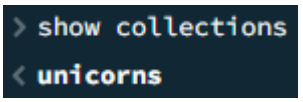


4) Получите список доступных БД



5) Создайте коллекцию unicorns, вставив в неё документ.

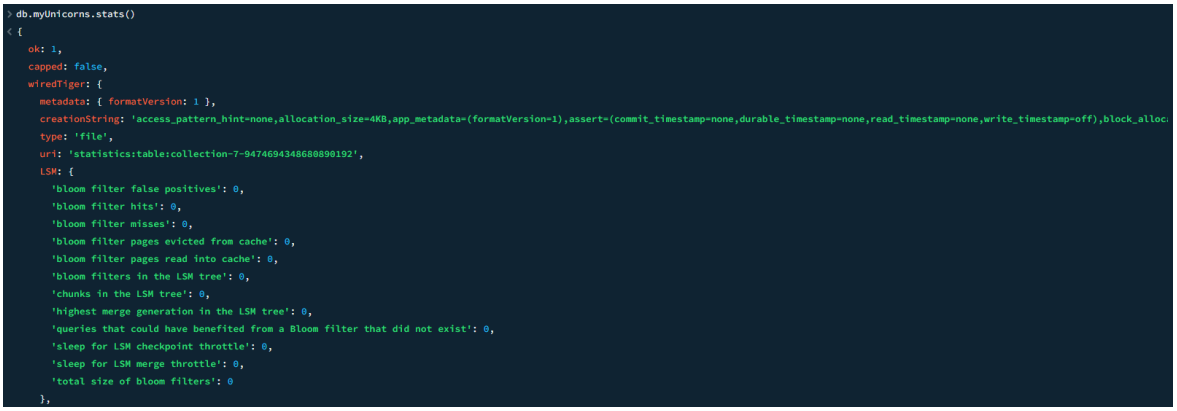
6) Просмотрите список текущих коллекций.



7) Переименуйте коллекцию unicorns.

```
> db.unicorns.renameCollection('myUnicorns')
< { ok: 1 }
```

8) Просмотрите статистику коллекции.



9) Удалите коллекцию.

```
> db.myUnicorns.drop()
< true
```

10) Удалите БД learn

```
> db.dropDatabase()
< { ok: 1, dropped: 'learn' }
```

2 CRUD-операции в СУБД mongodb. вставка данных. выборка данных

2.1 Вставка документов в коллекцию

1) Создайте базу данных learn.

Уже есть👍

2) Заполните коллекцию единорогов unicorns…

```
> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
  db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
  db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
  db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
  db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
  db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
  db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
  db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
  db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
  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'});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('68360ff94bc08c8985bdeb54')
    }
  }
```

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

```
> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
< {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
> db.unicorns.insertOne(document)
< {
    acknowledged: true,
    insertedId: ObjectId('683610634bc08c8985bdeb55')
  }
```

4) Проверьте содержимое коллекции с помощью метода find.

```
> db.unicorns.find()
< {
    _id: ObjectId('68360ff94bc08c8985bdeb4a'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('68360ff94bc08c8985bdeb4b'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('68360ff94bc08c8985bdeb4c'),
    name: 'Unicrom',
    loves: [
      'energon',
      'redbull'
    ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
  {
    _id: ObjectId('68360ff94bc08c8985bdeb4d'),
    name: 'Roooooodles',
```

2.2 Выборка данных из БД

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

```
> db.unicorns.find({gender:'f'}).limit(3).sort({name:1})
< {
    _id: ObjectId('68360ff94bc08c8985bdeb4b'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('68360ff94bc08c8985bdeb4f'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
    _id: ObjectId('68360ff94bc08c8985bdeb52'),
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
```

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('68360ff94bc08c8985bdeb4b'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

```
> db.unicorns.find({gender:'f', loves:'carrot'}).limit(1)
< {
    _id: ObjectId('68360ff94bc08c8985bdeb4b'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

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

```
> db.unicorns.find({gender:'m'}, {loves:0})
< {
    _id: ObjectId('68360ff94bc08c8985bdeb4a'),
    name: 'Horny',
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('68360ff94bc08c8985bdeb4c'),
    name: 'Unicrom',
    weight: 984,
    gender: 'm',
    vampires: 182
  }
  {
    _id: ObjectId('68360ff94bc08c8985bdeb4d'),
    name: 'Roooooodles',
    weight: 575,
    gender: 'm',
    vampires: 99
  }
  {
    _id: ObjectId('68360ff94bc08c8985bdeb50'),
    name: 'Kenny',
    weight: 690,
    gender: 'm',
    vampires: 39
  }
  {
    _id: ObjectId('68360ff94bc08c8985bdeb51'),
    name: 'Raleigh',
    weight: 421,
    gender: 'm',
    vampires: 2
  }
```

4) Вывести список единорогов в обратном порядке добавления.

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

```
> db.unicorns.find({}, {_id:0, loves:{$slice:[0, 1]}})
< {
    name: 'Horny',
    loves: [
      'carrot'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    name: 'Aurora',
    loves: [
      'carrot'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    name: 'Unicrom',
    loves: [
      'energon'
    ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
  {
    name: 'Roooooodles',
    loves: [
      'apple'
    ],
    weight: 575,
    gender: 'm',
    vampires: 99
  }
```

2.3 Логические операторы

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

```
> db.unicorns.find({gender:'f', weight:{$gt:500,$lt:700}}, {_id:0})
< {
    name: 'Solnara',
    loves: [
      'apple',
      'carrot',
      'chocolate'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
  {
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
```

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

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

3) Найти всех единорогов, не имеющих ключ vampires.

```
> db.unicorns.find({vampires:{$exists:0}})
< {
    _id: ObjectId('6836f2411fd1ee80412e7585'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

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

```
> db.unicorns.find({gender:'m'}, {_id:0, name:1, loves:{$slice:1}}).sort({name:1})
< {
    name: 'Dunx',
    loves: [
      'grape'
    ]
  }
  {
    name: 'Horny',
    loves: [
      'carrot'
    ]
  }
  {
    name: 'Kenny',
    loves: [
      'grape'
    ]
  }
  {
    name: 'Pilot',
    loves: [
      'apple'
    ]
  }
```

3 Запросы к базе данных mongodb. Выборка данных. вложенные объекты. Использование курсоров. Агрегированные запросы. Изменение данных

3.1 Запрос к вложенным объектам

1) Создайте коллекцию towns, включающую следующие документы…

Создал👍

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'}, {_id:0, name:1, mayor:1})
< {
    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}}, {_id:0, name:1, mayor:1})
< {
    name: 'Punxsutawney ',
    mayor: {
      name: 'Jim Wehrle'
    }
  }
```

4) Сформировать функцию для вывода списка самцов единорогов.

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

6) Вывести результат, используя forEach.

```
> fn = function() { return this.gender=='m'; }
  var cursor = db.unicorns.find({$where:fn}).sort({name:1}).limit(2);
  cursor.forEach(function(obj) {
     print(obj);
  })
< {
    _id: ObjectId('6836f25e1fd1ee80412e7586'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
< {
    _id: ObjectId('6836f2411fd1ee80412e757b'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
```

3.2 Агрегированные запросы

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

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

2) Вывести список предпочтений.

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

3) Подсчитать количество особей единорогов обоих полов.

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

3.3 Редактирование данных

1) Выполнить команду: db.unicorns.insertOne({ name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})

2) Проверить содержимое коллекции unicorns.

```
{
  _id: ObjectId('6836f5c01fd1ee80412e758a'),
  name: 'Barny',
  loves: [
    'grape'
  ],
  weight: 340,
  gender: 'm'
}
```

3) Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вампира.

```
{
  _id: ObjectId('6836f2411fd1ee80412e7580'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 800,
  gender: 'f',
  vampires: 51
}
```

4) Для самца единорога Raleigh внести изменения в БД: теперь он любит ред булл.

5) Всем самцам единорогов увеличить количество убитых вампиров на 5.

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

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

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

3.4 Удаление данных из коллекции

1) Создайте коллекцию towns, включающую следующие документы…

Создано👍

2) Удалите документы с беспартийными мэрами.

```
> db.towns.find()
< {
    _id: ObjectId('6836f7381fd1ee80412e758c'),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [
      'status of liberty',
      'food'
    ],
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
  {
    _id: ObjectId('6836f7381fd1ee80412e758d'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [
      'beer',
      'food'
    ],
    mayor: {
      name: 'Sam Adams',
      party: 'D'
    }
  }
```

3) Проверьте содержание коллекции.

Проверил

4) Очистите коллекцию.

5) Просмотрите список доступных коллекций.

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

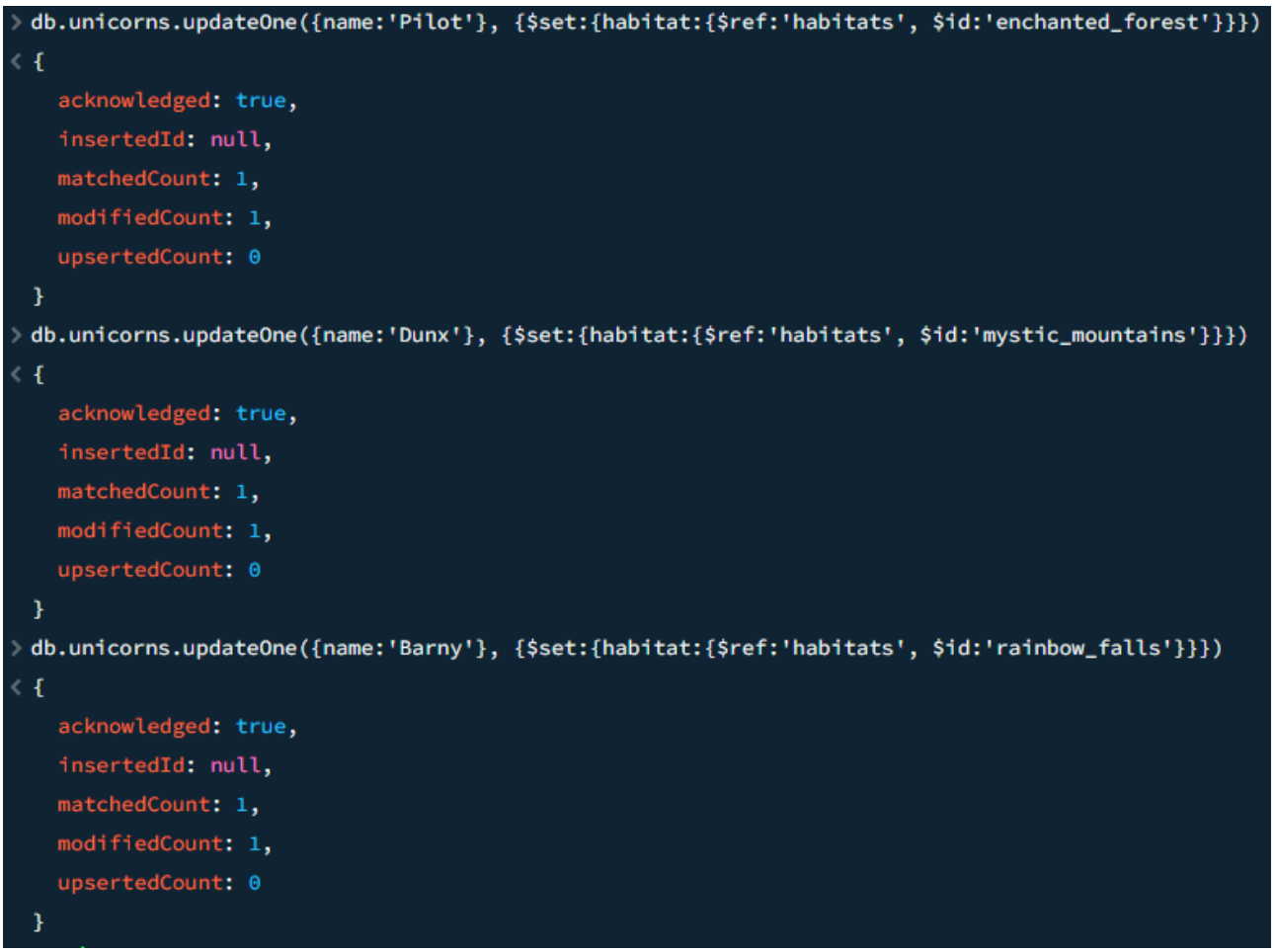
4 Ссылки и работа с индексами в базе данных mongodb

4.1 Ссылки в БД

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

```
> db.habitats.insertMany([
    {
      _id: "enchanted_forest",
      fullName: "Зачарованный лес",
      description: "Густой, мистический лес, где деревья светятся мягким светом и растут волшебные грибы. Здесь единороги чувствуют себя в безопасности."
    },
    {
      _id: "crystal_glade",
      fullName: "Хрустальная поляна",
      description: "Открытая поляна с кристально чистыми озерами и цветами, которые переливаются всеми цветами радуги. Любимое место единорогов для отдыха."
    },
    {
      _id: "mystic_mountains",
      fullName: "Мистические горы",
      description: "Высокие горы с серебряными вершинами, где воздух наполнен магией. Единороги приходят сюда для медитации."
    },
    {
      _id: "silver_valley",
      fullName: "Серебряная долина",
      description: "Плодородная долина, где трава переливается серебряным светом. Основное место сбора единорогов по ночам."
    },
    {
      _id: "rainbow_falls",
      fullName: "Радужные водопады",
      description: "Водопады, создающие постоянные радуги в воздухе. Единороги приходят сюда, чтобы пить волшебную воду."
    }
  ])
```

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



3) Проверьте содержание коллекции единорогов

4)

```
{
  _id: ObjectId('6836f25e1fd1ee80412e7586'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 170,
  habitat: DBRef('habitats', 'mystic_mountains')
}
{
  _id: ObjectId('6836f5c01fd1ee80412e758a'),
  name: 'Barny',
  loves: [
    'grape'
  ],
  weight: 340,
  gender: 'm',
  vampires: 5,
  habitat: DBRef('habitats', 'rainbow_falls')
}
```

4.2 Настройка индексов

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.insert({name:'Barny'})
✖ ▸ MongoBulkWriteError: E11000 duplicate key error collection: test.unicorns index: name_1 dup key: { name: "Barny" }
```

4.3 Управление индексами

1) Получите информацию о всех индексах коллекции unicorns.

```
> 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 }
```

3) Попытайтесь удалить индекс для идентификатора.

```
> db.unicorns.dropIndex("_id")
✖ ▸ MongoServerError[IndexNotFound]: index not found with name [_id]
> db.unicorns.dropIndex("_id_")
✖ ▸ MongoServerError[InvalidOptions]: cannot drop _id index
```

4.4 План запроса

1) Создайте объемную коллекцию numbers, задействовав курсор: for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}

Создал

2) Выберите последних четыре документа.

```
> db.numbers.find().skip(100000-4)
< {
    _id: ObjectId('6837012a1fd1ee80412ffc2d'),
    value: 99996
  }
  {
    _id: ObjectId('6837012a1fd1ee80412ffc2e'),
    value: 99997
  }
  {
    _id: ObjectId('6837012a1fd1ee80412ffc2f'),
    value: 99998
  }
  {
    _id: ObjectId('6837012a1fd1ee80412ffc30'),
    value: 99999
  }
```

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

```
executionTimeMillis: 28,
```

без сортировки

```
executionTimeMillis: 75,
```

с сортировкой

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.

```
executionTimeMillis: 29,
```

без сортировки

```
executionTimeMillis: 44,
```

с сортировкой

7) Проанализируйте план выполнения запроса с установленным индексом.

Если в запросе используется сортировка, то запрос с индексом будет быстрее. Иначе ничего не поменяется.

8) Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

Если в запросе используется сортировка, то запрос с индексом будет быстрее. Иначе ничего не поменяется.

Выводы

В ходе выполнения лабораторной работы я научился работать с СУБД mongoDB, выполнять в ней запросы и увидел, как индексы ускоряют запросы.