Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

«Работа с БД в СУБД MongoDB»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Крамарь Кирилл Александрович
Факультет прикладной информатики
Группа К3239
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии
2023 **Преподаватель** Говорова Марина Михайловна

Санкт-Петербург
2024/2025

СОДЕРЖАНИЕ

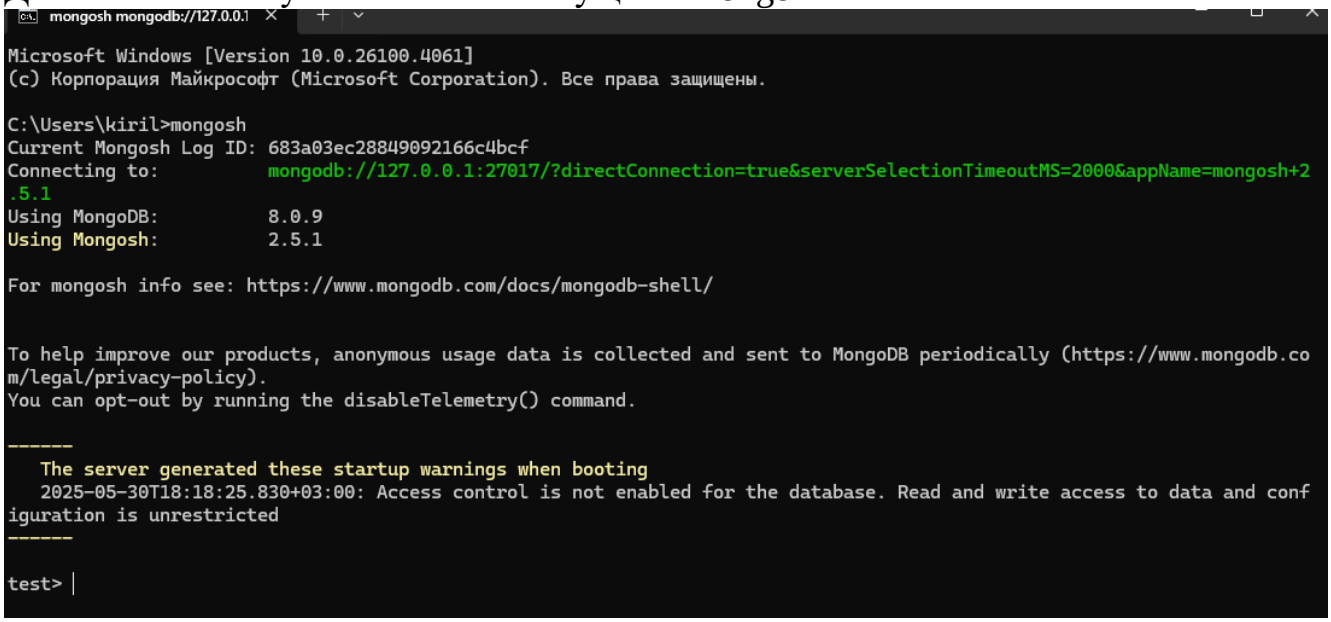
Стр.

1 Цель работы

Целью работы является получение практического опыта в выполнении CRUD-операций, работе с вложенными структурами в коллекциях MongoDB, применении агрегатных функций и изменении данных, а также освоении механизмов ссылок и индексирования в базе данных MongoDB.

2. Выполнение ЛР6
2.0 Установка и запуск

Для начала была установлена и запущена MongoDB



2.1.1 Задание

Заполним коллекцию unicorns и вставим в коллекцию документ, каждый раз будем проверять полноту базы данных.

```
learn;> db.unicorns.insertMany([
...    {name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63},
...    {name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43},
...    {name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182},
...    {name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99},
...    {name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80},
...    {name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40},
...    {name:'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39},
...    {name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2},
...    {name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33},
...    {name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54},
...    {name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'}
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683a0801288490921666c4bd0'),
    '1': ObjectId('683a0801288490921666c4bd1'),
    '2': ObjectId('683a0801288490921666c4bd2'),
    '3': ObjectId('683a0801288490921666c4bd3'),
    '4': ObjectId('683a0801288490921666c4bd4'),
    '5': ObjectId('683a0801288490921666c4bd5'),
    '6': ObjectId('683a0801288490921666c4bd6'),
    '7': ObjectId('683a0801288490921666c4bd7'),
    '8': ObjectId('683a0801288490921666c4bd8'),
    '9': ObjectId('683a0801288490921666c4bd9'),
    '10': ObjectId('683a0801288490921666c4bda')
  }
}
learn;> db.unicorns.find().pretty()
[
  {
    _id: ObjectId('683a0801288490921666c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('683a0801288490921666c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683a0801288490921666c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683a0801288490921666c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('683a0801288490921666c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683a0801288490921666c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
```

```
]
learn;> var document = {
...    name: 'Dunx',
...    loves: ['grape', 'watermelon'],
...    weight: 704,
...    gender: 'm',
...    vampires: 165
... };
...
... db.unicorns.insertOne(document);
...
{
  acknowledged: true,
  insertedId: ObjectId('683a085628849092166c4bdb')
}
learn;>
```

```
  {
    _id: ObjectId('683a080128849092166c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('683a085628849092166c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
learn;>
```

2.2.1 Задание

Выведем список, состоящий из самцов, первых 3-ёх самок и отсортируем его:

```
learn> db.unicorns.find({gender: "m"})
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861e1'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e3'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e4'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
learn> db.unicorns.find({gender: "f"}).limit(3)
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861e2'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e5'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e6'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
]
```

Выведу самку, которая любит морковь:

2.2.2 Задание

Выведем список самцов без loves и без указания gender

2.2.3 Задание

Выведем список в обратном порядке как мы добавляли значения в него:

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('68375259ffe7cdce97d861ec'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861eb'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
```

2.2.4 Задание

Выведем список с названием первого loves, при этом исключим идентификатор

```
learn> db.unicorns.find({}, {loves: {$slice: 1}, _id: 0})
[
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Aurora',
    loves: [ 'carrot' ],
    weight: 450,
    gender: 'f',
    vampires: 43
```

2.3.1 Задание

Выведем список самок с весом от 500кг до 700кг. Также исключим идентификатор:

```
learn;> db.unicorns.find({gender: "f", weight: {$gte: 500, $lte: 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
  },
  {
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
```

2.3.2 Задание:

Вывод списка самцов весом от 500 кг и с loves grape и lemon, идентификатор дополнительно исключим.

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

2.3.3 Задание:

Выведем всех единорогов не vampires

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861eb'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

2.3.4 Задание:

Выведем упорядоченный список имен самцов с информацией об их loves

```
learn> db.unicorns.find({gender: "m"}, {name: 1, loves: {$slice: 1}}).sort({name: 1})
[
  {
    _id: ObjectId('68375259ffe7cdce97d861ec'),
    name: 'Dunx',
    loves: [ 'grape' ]
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e1'),
    name: 'Horny',
    loves: [ 'carrot' ]
  },
```

3.1.1 Задание:

Создадим новую коллекцию. В ней будут мэры городов США, сформируем запросы на возвращение списка городов с независимыми мэрами, а также списка беспартийных мэров

```
learn;> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     populatiuon: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: [""],
...     mayor: { name: "Jim Wehrle" }
...   },
...   {
...     name: "New York",
...     populatiuon: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["status of liberty", "food"],
...     mayor: { name: "Michael Bloomberg", party: "I" }
...   },
...   {
...     name: "Portland",
...     populatiuon: 528000,
...     last_sensus: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: { name: "Sam Adams", party: "D" }
...   }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683a0bdd28849092166c4bdc'),
    '1': ObjectId('683a0bdd28849092166c4bdd'),
    '2': ObjectId('683a0bdd28849092166c4bde')
  }
}
learn;>
```

2

```
learn;> db.towns.find({ "mayor.party": "I" }, { name: 1, mayor: 1 })
[
  {
    _id: ObjectId('683a0bdd28849092166c4bdd'),
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
```

3

```
learn;> db.towns.find({ "mayor.party": { $exists: false } }, { name: 1, mayor: 1 })
[
  {
    _id: ObjectId('683a0bdd28849092166c4bdc'),
    name: 'Punxsutawney',
    mayor: { name: 'Jim Wehrle' }
  }
]
```

3.1.2 Задание:

С использование JavaScript напишем функцию, нацеленную на создание курсора и вывод результата

```
learn;> var cursor = db.unicorns.find({ gender: "m" }).sort({ name: 1 }).limit(2);
... cursor.forEach(function(unicorn) {
...   print(unicorn.name);
... });
...
Dunx
Horny

learn;>
```

3.2.1 Задание:

Подсчитаем самок с требуемым весом

```
learn> db.unicorns.find({gender: "f", weight: {$gte: 500, $lte: 600}}).count(true)
2
```

3.2.2 Задание:

Выведем список предпочтений:

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

3.2.3 Задание:

Выведем общее количество особей обоих полов

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

3.3.1 Задание:

Выполненим команду save в инструкции, зафиксируем ошибку. Вместо save необходим insertOne:

```
learn> db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
TypeError: db.unicorns.save is not a function
learn> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedId: ObjectId('683789b963b9c722c1d861ef')
}
```

Проверим:

```
  _id: ObjectId('683789b963b9c722c1d861ef'),
  name: 'Barny',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm'
```

Теперь всё работает.

3.3.2 Задание:

Сделаем изменение для Ayna и проверим качество изменений

Всё работает.

3.3.3 Задание:

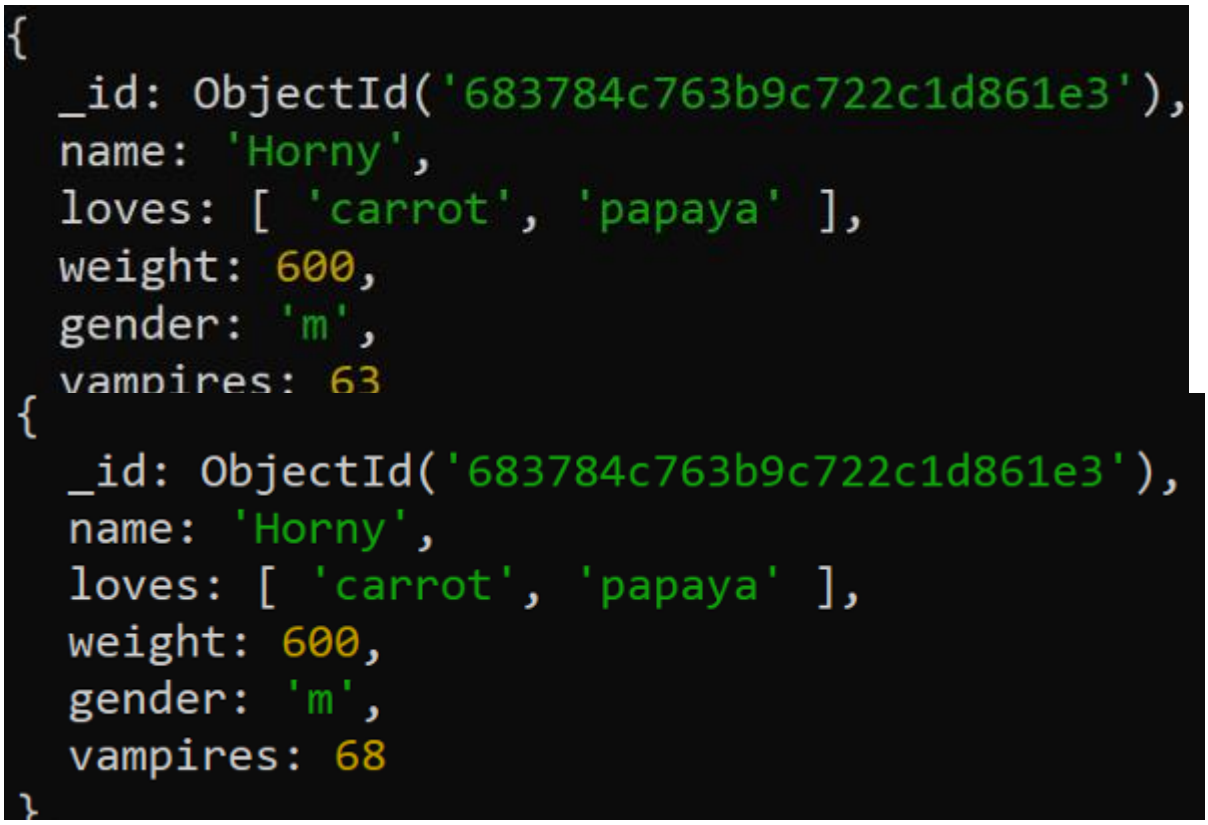
Согласно инструкции, добавим в loves для Raleigh

```
learn> db.unicorns.updateOne(
...    { name: "Raleigh",},
...    { $push: { loves: "redbull" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Raleigh" })
[
  {
    _id: ObjectId('683784c763b9c722c1d861ea'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
```

3.3.4 Задание:

Сделаем большее количество особей вампирами

```
learn> db.unicorns.updateMany(
...    { gender: "m" },
...    { $inc: { vampires: 5 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
```



Сравнение показывает наличие изменений

3.3.5 Задание:

Внесем изменения в Портленд

```
{
  _id: ObjectId('683781b463b9c722c1d861e2'),
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
  famous_for: [ 'beer', 'food' ],
  mayor: { name: 'Sam Adams' }
}
```

3.3.6 Задание:

Изменяем loves у Pilot

```
learn> db.unicorns.updateOne(
...    { name: "Pilot"},
...    { $addToSet: { loves: "chocolate" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Pilot" })
[
  {
    _id: ObjectId('683784c763b9c722c1d861ec'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

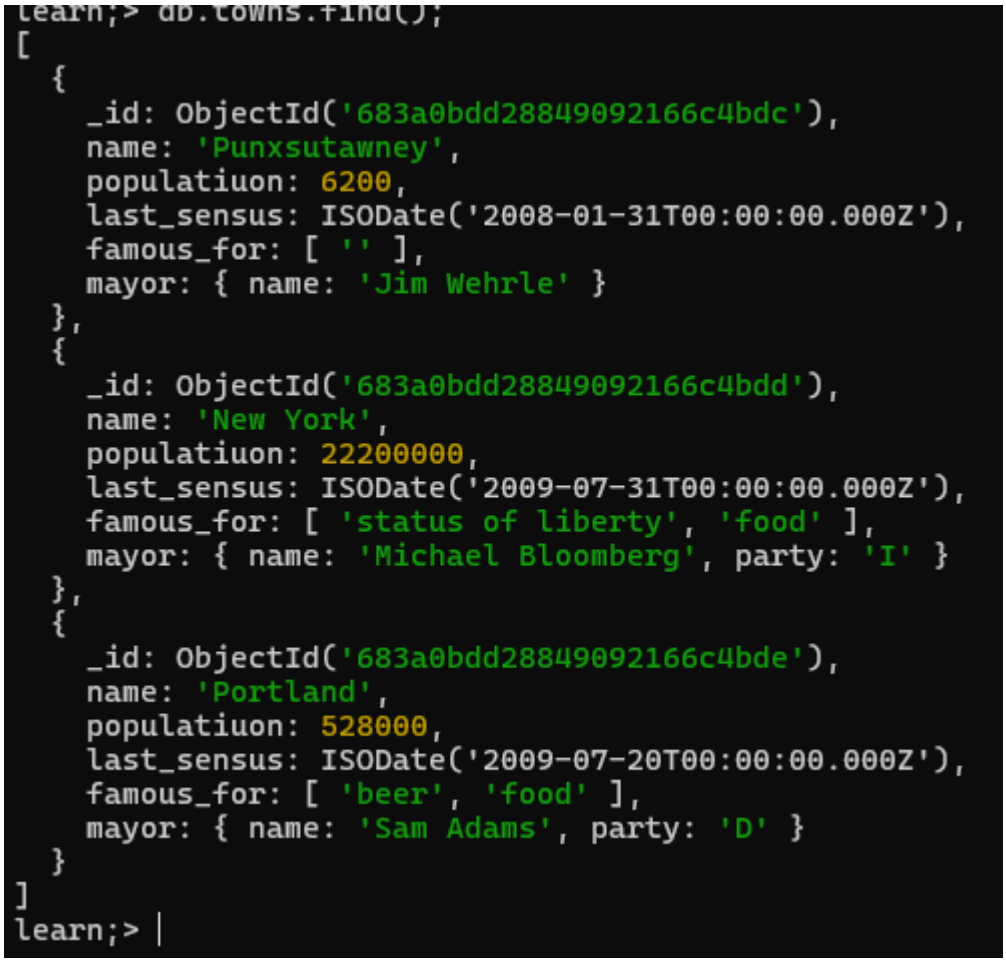
3.3.7 Задание:

Внесем изменения в loves для Aurora

3.4.1 Задание:

Удалим мэров без партии.

Список городов:



Произвели удаление:

```
    _id: ObjectId('683a0bdd28849092166c4bde'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn;> db.towns.deleteMany({ "mayor.party": { $exists: false } });
{ acknowledged: true, deletedCount: 1 }
learn;> db.towns.find();
[
  {
    _id: ObjectId('683a0bdd28849092166c4bdd'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('683a0bdd28849092166c4bde'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn;>
```

Чистим коллецию

```
earn> db.towns.deleteMany({})
 acknowledged: true, deletedCount: 2 }
```

Проверяем:

```
learn;> show collections
towns
unicorns
```

4.1.1 Задание:

Создадим коллекцию зон обитания. Добавим их к Единорогам

```
earn;> db.habitats.insertMany([
...   { _id: 1, name: "Forest" },
...   { _id: 2, name: "Mountains" },
...   { _id: 3, name: "Swamp" }
... ]);
...
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2': 3 } }
earn;>
```

```
learn;> db.habitats.insertMany([
...    { _id: 1, name: "Forest" },
...    { _id: 2, name: "Mountains" },
...    { _id: 3, name: "Swamp" }
... ]);
...
{ acknowledged: true, insertedIds: { '0': 1, '1': 2, '2'
learn;> db.unicorns.updateOne(
...    { name: "Horny" },
...    { $set: { habitat_id: 1 } }
... );
...
... db.unicorns.updateOne(
...    { name: "Aurora" },
...    { $set: { habitat_id: 2 } }
... );
...
... db.unicorns.updateOne(
...    { name: "Unicrom" },
...    { $set: { habitat_id: 3 } }
... );
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn;>
```

```
learn;> db.unicorns.find(
...    { name: { $in: ["Horny", "Aurora", "Unicrom"] } },
...    { name: 1, habitat_id: 1, _id: 0 }
... );
...
[
  { name: 'Horny', habitat_id: 1 },
  { name: 'Aurora', habitat_id: 2 },
  { name: 'Unicrom', habitat_id: 3 }
]
learn;>
```

4.2.1 Задание:

Осуществим проверку возможности задать индекс:

```
learn;> db.unicorns.createIndex({ name: 1 }, { unique: true })
name_1
```

4.3.1 Задание:

1. Получим информацию про индексы
2. Удалим все, кроме идентификаторного
3. Попытаемся удалить индекс _id_

```
learn;> db.unicorns.find(
...   { name: { $in: ["Horny", "Aurora", "Unicrom"] } },
...   { name: 1, habitat_id: 1, _id: 0 }
... );
...
[
  { name: 'Horny', habitat_id: 1 },
  { name: 'Aurora', habitat_id: 2 },
  { name: 'Unicrom', habitat_id: 3 }
]
learn;> db.unicorns.createIndex({ name: 1 }, { unique: true })
name_1
learn;> db.unicorns.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
learn;> db.unicorns.dropIndexes()
{
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
learn;>
```

Неудачная попытка удаления:

```
learn> db.unicorns.dropIndex("_id_")
MongoServerError[InvalidOptions]: cannot drop _id index
```

4.4.1 Задание:

```
learn;> for (let i = 0; i < 10000; i++) {
...   db.bigdata.insertOne({
...     number: i,
...     isEven: i % 2 === 0,
...     randomText: "value_" + i
...   });
... }
...
{
  acknowledged: true,
  insertedId: ObjectId('683a10dd28849092166cd2f0')
}
```

Проверка времени на 10 документов:

```
learn;> db.bigdata.find({
...   number: { $in: [1, 10, 100, 1000, 2000, 3000, 4000, 5000, 6000, 7000] }
... }).explain("executionStats");
...
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn;.bigdata',
    parsedQuery: {
      number: {
        '$in': [
             1,   10,  100,
          1000, 2000, 3000,
          4000, 5000, 6000,
          7000
        ]
      }
    },
    indexFilterSet: false,
    queryHash: '698418AE',
    planCacheShapeHash: '698418AE',
    planCacheKey: '332B82AE',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: {
        number: {
          '$in': [
               1,   10,  100,
```

```
learn;> db.bigdata.find({ number: 9999 }).explain("executionStats");
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn;.bigdata',
    parsedQuery: { number: { '$eq': 9999 } },
    indexFilterSet: false,
    queryHash: 'F966BACD',
    planCacheShapeHash: 'F966BACD',
    planCacheKey: 'A62581B9',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { number: { '$eq': 9999 } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 3,
    totalKeysExamined: 0,
    totalDocsExamined: 17611,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
```


'$in': [
1, 10, 100,
1000, 2000, 3000,
4000, 5000, 6000,
7000
]
}
},
direction: 'forward'
},
rejectedPlans: []
},
executionStats: {
executionSuccess: true,
nReturned: 20,
executionTimeMillis: 4,
totalKeysExamined: 0,
totalDocsExamined: 17611,
executionStages: {
isCached: false,
stage: 'COLLSCAN',
filter: {
number: {
'$in': [
1, 10, 100,
1000, 2000, 3000,
4000, 5000, 6000,
7000
]
}
},
nReturned: 20,
executionTimeMillisEstimate: 0,
works: 17612,
advanced: 20,
needTime: 17591,
needYield: 0,
saveState: 0,
restoreState: 0,
isEOF: 1,
direction: 'forward',

```
    isEOF: 1,
    direction: 'forward',
    docsExamined: 17611
  }
},
queryShapeHash: 'CDCDDE3F64F5CDE960A5BD231C8F962BD46BFF1018A245AE570600E8B70680FA',
command: {
  find: 'bigdata',
  filter: {
    number: {
      '$in': [
           1,   10,  100,
        1000, 2000, 3000,
        4000, 5000, 6000,
        7000
      ]
    }
  },
  '$db': 'learn;'
},
serverInfo: {
  host: 'Kirill-PC',
  port: 27017,
  version: '8.0.9',
  gitVersion: 'f882ef816d531ecfbb593843e4c554fda90ca416'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted',
  internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
ok: 1
```

Теперь сделаем индекс:

```
learn;> db.bigdata.createIndex({ number: 1 });
number_1
```

Проверим индексы:

```
ReferenceError: bers is not defined
learn;> db.numbers.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn;>
```

Посмотрим скорости:

Запрос с индексами выполнился действительно быстрее.

А теперь правильное выполнение для 4 индексов:
До индексов:

```
learn;> db.bigdata.find({
...   number: { $in: [111, 2222, 3333, 44444] }
... }).explain("executionStats");
...
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn;.bigdata',
    parsedQuery: { number: { '$in': [ 111, 2222, 3333, 44444 ] } },
    indexFilterSet: false,
    queryHash: '698418AE',
    planCacheShapeHash: '698418AE',
    planCacheKey: '8E634D2B',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { number: 1 },
        indexName: 'number_1',
        isMultiKey: false,
        multiKeyPaths: { number: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
          number: [
            '[111, 111]',
            '[2222, 2222]',
            '[3333, 3333]',
            '[44444, 44444]'
          ]
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 8,
    executionTimeMillis: 0,
    totalKeysExamined: 12,
```


executionStats: {
executionSuccess: true,
nReturned: 8,
executionTimeMillis: 0,
totalKeysExamined: 12,
totalDocsExamined: 8,
executionStages: {
isCached: false,
stage: 'FETCH',
nReturned: 8,
executionTimeMillisEstimate: 0,
works: 12,
advanced: 8,
needTime: 3,
needYield: 0,
saveState: 0,
restoreState: 0,
isEOF: 1,
docsExamined: 8,
alreadyHasObj: 0,
inputStage: {
stage: 'IXSCAN',
nReturned: 8,
executionTimeMillisEstimate: 0,
works: 12,
advanced: 8,
needTime: 3,
needYield: 0,
saveState: 0,
restoreState: 0,
isEOF: 1,
keyPattern: { number: 1 },
indexName: 'number_1',
isMultiKey: false,
multiKeyPaths: { number: [] },
isUnique: false,
isSparse: false,
isPartial: false,
indexVersion: 2,
direction: 'forward',
indexBounds: {
number: [
'[111, 111]',
'[2222, 2222]',
'[3333, 3333]',
'[44444, 44444]'
]
},
keysExamined: 12,
seeks: 4,
dupsTested: 0,
dupsDropped: 0
}
}
},
queryShapeHash: 'CDCDDE3F64F5CDE960A5BD231C8F962BD46BFF1018A245AE570600E8B70680FA',

Делаем индекс:

```
learn;> db.bigdata.createIndex({ number: 1 });
number_1
learn;>
```

Проверяем с индексом:

Таким образом, заметно, что индекс работает.

Выводы

В процессе выполнения лабораторной работы я закрепил практические навыки работы с операциями создания, чтения, обновления и удаления данных (CRUD), освоил обращение к вложенным структурам в коллекциях MongoDB, выполнил агрегационные запросы и модификацию данных, а также поработал с индексами и связями между документами в базе данных MongoDB.