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»**

Автор: Зотов М.Д.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

Цель:

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.

Выполнение работы

Практическое задание 2.1.1:

1. *Создайте базу данных learn.*

```
test> use learn
switched to db learn
```

2. *Заполните коллекцию единорогов unicorns:*

```
learn> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66018659a7040d3c1ed14a0e') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66018661a7040d3c1ed14a0f') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66018669a7040d3c1ed14a10') }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66018671a7040d3c1ed14a11') }
}
learn> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6601867ca7040d3c1ed14a12') }
}
learn> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66018684a7040d3c1ed14a13') }
}
learn> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6601868ba7040d3c1ed14a14') }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66018692a7040d3c1ed14a15') }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6601869aa7040d3c1ed14a16') }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660186a0a7040d3c1ed14a17') }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660186aba7040d3c1ed14a18') }
}
```

3. *Используя второй способ, вставьте в коллекцию единорогов документ*

```
learn> document = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660186eba7040d3c1ed14a19') }
}
```

4. *Проверьте содержимое коллекции с помощью метода find:*

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('66018659a7040d3c1ed14a0e'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('66018661a7040d3c1ed14a0f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
learn>
  {
    _id: ObjectId('66018669a7040d3c1ed14a10'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('66018671a7040d3c1ed14a11'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6601867ca7040d3c1ed14a12'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('66018684a7040d3c1ed14a13'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('6601868ba7040d3c1ed14a14'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
```

Практическое задание 2.2.1:

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

```
learn> db.unicorns.find({gender: "m"}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('660186eba7040d3c1ed14a19'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('66018659a7040d3c1ed14a0e'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6601868ba7040d3c1ed14a14'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

```
learn> db.unicorns.find({gender: "f"}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('66018661a7040d3c1ed14a0f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('66018684a7040d3c1ed14a13'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('6601869aa7040d3c1ed14a16'),
    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.*

```
learn> db.unicorns.findOne({gender: "f", loves: "carrot"})
{
  _id: ObjectId('66018661a7040d3c1ed14a0f'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

```
learn> db.unicorns.find({gender: "f", loves: "carrot"}).limit(1)
[
  {
    _id: ObjectId('66018661a7040d3c1ed14a0f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 2.2.2:

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

```
learn> db.unicorns.find({gender: "m"}, {loves: 0, gender: 0}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('660186eba7040d3c1ed14a19'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('66018659a7040d3c1ed14a0e'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('6601868ba7040d3c1ed14a14'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
]
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

Практическое задание 2.2.4:

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

```
learn> db.unicorns.find({}, {_id: 0, loves: {$slice: 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
  },
  {
    name: 'Solnara',
    loves: [ 'apple' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    name: 'Ayna',
    loves: [ 'strawberry' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    name: 'Kenny',
    loves: [ 'grape' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    name: 'Raleigh',
    loves: [ 'apple' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
```

Практическое задание 2.3.1:

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

```
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'
  }
]
```

Практическое задание 2.3.2:

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

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

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ `vampires`.

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('660186aba7040d3c1ed14a18'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.4:

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

```
learn> db.unicorns.find({gender: "m"}, {_id: 0, loves: {$slice: 1}, weight: 0, gender: 0, vampires: 0})
[
  { name: 'Horny', loves: [ 'carrot' ] },
  { name: 'Unicrom', loves: [ 'energon' ] },
  { name: 'Roooooodles', loves: [ 'apple' ] },
  { name: 'Kenny', loves: [ 'grape' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Pilot', loves: [ 'apple' ] },
  { name: 'Dunx', loves: [ 'grape' ] }
]
```

Практическое задание 3.1.1:

1) Создайте коллекцию towns, включающую следующие документы:

```
learn> db.createCollection("towns")
{ ok: 1 }
learn> db.towns.insert({name: "Punxsutawney ", populatiuon: 6200, last_sensus: ISODate("2008-01-31"), famous_for: [""], mayor: {name: "Jim Wehrle"}})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660198aba7040d3c1ed14a1a') }
}
learn> db.towns.insert({name: "New York", populatiuon: 22200000, last_sensus: ISODate("2009-07-31"), famous_for: ["status of liberty", "food"], mayor: {name: "Michael Bloomberg", party: "I"}})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660198c2a7040d3c1ed14a1b') }
}
learn> db.towns.insert({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('660198d6a7040d3c1ed14a1c') }
}
```

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

```
learn> db.towns.find({"mayor.party": "I"}, {_id: 0, populatiuon: 0, last_sensus: 0, ISODate: 0, famous_for: 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.*

```
learn> db.towns.find({"mayor.party": {$exists: false}}, {_id: 0, populatiuon: 0, last_sensus: 0, ISODate: 0, famous_for: 0})
[ { name: 'Punxsutawney ', mayor: { name: 'Jim Wehrle' } } ]
```

Практическое задание 3.1.2:

Сформировать функцию для вывода списка самцов единорогов.
Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.
Вывести результат, используя forEach.

```
learn> function getMaleUnicorns() {
... var cursor = db.unicorns.find({gender: "m"}).sort({name: 1}).limit(2);null
... cursor.forEach(function(unicorn) {
... print(unicorn);
... })};
[Function: getMaleUnicorns]
learn> getMaleUnicorns();
{
  _id: ObjectId('660186eba7040d3c1ed14a19'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('66018659a7040d3c1ed14a0e'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 3.2.2:

Вывести список предпочтений.

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

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

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

Практическое задание 3.3.1:

Выполнить команду:

```
> db.unicorns.save({name: 'Barny', loves: ['grape'],
weight: 340, gender: 'm'})
```

Проверить содержимое коллекции `unicorns`.

```
learn> db.unicorns.insertOne({name: 'Barny', loves: ['grape'],weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedId: ObjectId('6601eb2aa7040d3c1ed14a1d')
}
```

```
  },
  {
    _id: ObjectId('660186aba7040d3c1ed14a18'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('660186eba7040d3c1ed14a19'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6601eb2aa7040d3c1ed14a1d'),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
```

Практическое задание 3.3.2:

Для самки единорога `Ayna` *внести изменения в БД: теперь ее вес 800, она убила 51 вапмира. Проверить содержимое коллекции* `unicorns`.

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

```
  {
    _id: ObjectId('6601867ca7040d3c1ed14a12'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('66018684a7040d3c1ed14a13'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  },
  {
    _id: ObjectId('6601868ba7040d3c1ed14a14'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
```

Практическое задание 3.3.3:

Для самца единорога `Raleigh` *внести изменения в БД: теперь он любит рэдбул.*

Проверить содержимое коллекции `unicorns`.

```
learn> db.unicorns.updateOne({name: "Raleigh", gender: "m"}, {$set: {loves: [ 'apple', 'sugar', 'redbull' ]}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
  },
  {
    _id: ObjectId('66018692a7040d3c1ed14a15'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
```

Практическое задание 3.3.4:

Всем самцам единорогов увеличить количество убитых вапмиров на 5.

Проверить содержимое коллекции `unicorns`.

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

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('66018659a7040d3c1ed14a0e'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('66018661a7040d3c1ed14a0f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('66018669a7040d3c1ed14a10'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
```

Практическое задание 3.3.5:

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

Проверить содержимое коллекции `towns`.

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

```
  {
    _id: ObjectId('660198d6a7040d3c1ed14a1c'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
```

Практическое задание 3.3.6:

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

Проверить содержимое коллекции `unicorns`.

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

```
  {
    _id: ObjectId('660186a0a7040d3c1ed14a17'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  },
```

Практическое задание 3.3.7:

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

Проверить содержимое коллекции `unicorns`.

```
learn> db.unicorns.updateOne({name: "Aurora", gender: "f"}, {$push: {loves: {$each: [ 'sugar', 'lemons' ]}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
  {
    _id: ObjectId('66018661a7040d3c1ed14a0f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
```

Практическое задание 3.4.1:

Удалите документы с беспартийными мэрами.

```
learn> db.towns.remove({"mayor.party": {$exists: 0}})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.deleteMany({"mayor.party": {$exists: 0}})
{ acknowledged: true, deletedCount: 0 }
```

Проверьте содержание коллекции.

```
learn> db.towns.find()
[
  {
    _id: ObjectId('6601f4f5a7040d3c1ed14a1f'),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('6601f504a7040d3c1ed14a20'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

Очистите коллекцию.

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

Просмотрите список доступных коллекций.

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

Практическое задание 4.1.1:

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

```
learn> db.areas.find()
[
  {
    _id: 'Russia',
    name: 'Russian Federation',
    description: 'The best country.'
  },
  {
    _id: 'USA',
    name: 'United states of America',
    description: 'The most ordinary country'
  }
]
```

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

```
learn> db.unicorns.updateOne({name: "Nimue"}, {$set: {country:{$ref: "areas", $id: "USA"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne({name: "Pilot"}, {$set: {country:{$ref: "areas", $id: "Russia"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Проверьте содержание коллекции единорогов.

```
{
  _id: ObjectId('6601fb81a7040d3c1ed14a2a'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon' ],
  weight: 650,
  gender: 'm',
  vampires: 54,
  country: DBRef('areas', 'Russia')
},
{
  _id: ObjectId('6601fb85a7040d3c1ed14a2b'),
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f',
  country: DBRef('areas', 'USA')
},
```

Практическое задание 4.2.1:

Проверьте, можно ли задать для коллекции `unicorns` *индекс для ключа* `name` *с флагом* `unique`

```
learn> db.unicorns.ensureIndex({"name" : 1}, {"unique" : true})
[ 'name_1' ]: 80
```

Практическое задание 4.3.1:

Получите информацию о всех индексах коллекции `unicorns` .

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

Удалите все индексы, кроме индекса для идентификатора.

```
learn> db.unicorns.dropIndex("name_1")
{ nIndexesWas: 2, ok: 1 }
```

Попытайтесь удалить индекс для идентификатора.

```
learn> db.unicorns.dropIndex("_id")
MongoshInternalError[IndexNotFound]: index not found with name [_id]
```

Практическое задание 4.4.1:

1. *Создайте объемную коллекцию* `numbers`*, задействовав курсор:*

   ```
   for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
   ```

2. *Выберите последних четыре документа.*

   ```
   learn> db.numbers.find().sort({$natural: -1}).limit(4)
   [
     { _id: ObjectId('660200efa7040d3c1ed2d0cc'), value: 99999 },
     { _id: ObjectId('660200efa7040d3c1ed2d0cb'), value: 99998 },
     { _id: ObjectId('660200efa7040d3c1ed2d0ca'), value: 99997 },
     { _id: ObjectId('660200efa7040d3c1ed2d0c9'), value: 99996 }
   ]
   ```

3. *Проанализируйте план выполнения запроса 2. Сколько потребовалось временина выполнение запроса? (по значению параметра* `exectionTimeMillis`*)*

```
learn> db.numbers.find().sort({$natural: -1}).limit(4).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '8880B5AF',
    planCacheKey: '8880B5AF',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: { stage: 'COLLSCAN', direction: 'backward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 1,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
      stage: 'LIMIT',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      limitAmount: 4,
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 4,
        executionTimeMillisEstimate: 0,
        works: 4,
        advanced: 4,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 0,
        direction: 'backward',
        docsExamined: 4
      }
```

4. *Создайте индекс для ключа* `value`.

```
learn> db.numbers.ensureIndex({"values": 1})
[ 'values_1' ]
```

5. *Получите информацию о всех индексах коллекции* `numbres`.

```
learn> db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { values: 1 }, name: 'values_1' }
]
```

6. *Выполните запрос 2.*

```
learn> db.numbers.find().sort({$natural: -1}).limit(4)
[
  { _id: ObjectId('660200efa7040d3c1ed2d0cc'), value: 99999 },
  { _id: ObjectId('660200efa7040d3c1ed2d0cb'), value: 99998 },
  { _id: ObjectId('660200efa7040d3c1ed2d0ca'), value: 99997 },
  { _id: ObjectId('660200efa7040d3c1ed2d0c9'), value: 99996 }
]
```

7. *Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?*

```
learn> db.numbers.find().sort({$natural: -1}).limit(4).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '8880B5AF',
    planCacheKey: '8880B5AF',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: { stage: 'COLLSCAN', direction: 'backward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
      stage: 'LIMIT',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      limitAmount: 4,
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 4,
        executionTimeMillisEstimate: 0,
        works: 4,
        advanced: 4,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 0,
        direction: 'backward',
        docsExamined: 4
      }
```

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

В запросе с индексом значение ExecutionTimeMillis = 0, а в запросе без индекса значение - 1. Следовательно, индекс немного уменьшил время выполнения запроса. Скорее всего, при обработке более объемных данных, эта разница была бы намного заметнее.

Дополнительное задание:

Создайте новую базу данных MongoDB с именем "task10db". В этой базе данных создайте коллекцию "products".

```
learn> use task10db
switched to db task10db
task10db> db.createCollection("products")
{ ok: 1 }
```

Добавьте не менее 500 записей о продуктах в коллекцию (название и цена), используйте цикл. Название не обязательно должно быть осмысленным.

```
task10db> function generatePrice() {
... return Math.floor(Math.random() * 10000) + 1;
... }
[Function: generatePrice]
task10db> function generateName() {
... var adjectives = ["Juicy", "Fragrant", "Tasty", "Spicy", "Exquisite", "Piquant", "Satisfying", "Sharp", "Ten
nique", "Luxurious", "Eastern", "Refined", "Fruit", "Homemade", "Hot", "Honey", "Baked", "Tasty", "Fried", "Cris
... var nouns = ["Soup", "Salad", "Fish", "Steak", "Pasta", "Sandwich", "Minced meat", "Roll", "Pizza", "Curry",
s", "Kebab", "Macaroni", "Sushi", "Shawarma", "Omelet", "Borsch", "Buckwheat", "Cutlets", "Spaghetti", "Bread",
, "Pelmeni", "Tacos", "Mutton", "Shrimp", "Stroganoff", "Schnitzel", "Kebab"];
... var randomAdjectiveIndex = Math.floor(Math.random() * adjectives.length);
... var randomNounIndex = Math.floor(Math.random() * nouns.length);
... return adjectives[randomAdjectiveIndex] + " " + nouns[randomNounIndex];
... }
[Function: generateName]
task10db> function addProducts() {
... for (var i = 0; i < 600; i++) {
... var productName = generateName();
... var productPrice = generatePrice();
... db.products.insertOne({name: productName, price: productPrice});
... }}
[Function: addProducts]
task10db> addProducts()
```

Осуществите запросы поиска минимальной цены, максимальной цены и сортировки продуктов по цене.

```
task10db> db.products.aggregate({$group: {_id: null, minPrice: {$min: "$price"}}})
[ { _id: null, minPrice: 7 } ]
task10db> db.products.aggregate({$group: {_id: null, maxPrice: {$max: "$price"}}})
[ { _id: null, maxPrice: 9991 } ]
task10db> db.products.find().sort({price: 1})
[
  {
    _id: ObjectId('66020955a7040d3c1ed2d10d'),
    name: 'Unique Cheeseburger',
    price: 7
  },
  {
    _id: ObjectId('66020955a7040d3c1ed2d1a2'),
    name: 'Homemade Kebab',
    price: 28
  },
  {
    _id: ObjectId('66020954a7040d3c1ed2d0f8'),
    name: 'Refined Roll',
    price: 29
  },
  {
    _id: ObjectId('66020955a7040d3c1ed2d1e2'),
    name: 'Unique Salmon',
    price: 64
  },
  {
    _id: ObjectId('66020955a7040d3c1ed2d257'),
    name: 'Juicy Fish',
    price: 65
  },
  {
    _id: ObjectId('66020955a7040d3c1ed2d234'),
    name: 'Refined Kebab',
    price: 73
  },
  {
    _id: ObjectId('66020955a7040d3c1ed2d24f'),
    name: 'Homemade Borsch',
    price: 74
  },
  {
    _id: ObjectId('66020955a7040d3c1ed2d134'),
    name: 'Sweet Borsch',
    price: 92
  },
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 10,
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 2,
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 600,
  executionTimeMillis: 2,
```

Создайте индекс для цены.

```
task10db> db.products.ensureIndex({"price": 1})
[ 'price_1' ]
```

Выполните те же запросы. Сравните производительность до и после создания индексов.

```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 2,
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 1,
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 600,
  executionTimeMillis: 2,
```

Вывод:

В данной лабораторной работе я овладел практическими навыками работы с CRUD- операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации иизменения данных, со ссылками и индексами в базе данных MongoDB.