

**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** Клименков Владислав Максимович  
**Факультет** прикладной информатики  
**Группа** K3241  
**Направление подготовки** 09.03.03 Прикладная информатика  
**Образовательная программа** Мобильные и сетевые технологии 2023  
**Преподаватель** Говорова Марина Михайловна

Санкт-Петербург  
2025

## 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 Практическая часть

### Задание 2.1.1

#### Описание:

- 1) Создайте базу данных "learn".
- 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'});
...
```

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

```
...
{name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm',
vampires: 165}
...
```

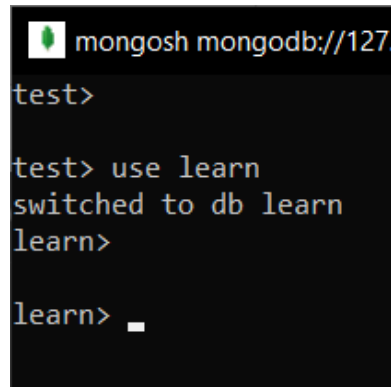
#### 4) Проверьте содержимое коллекции с помощью метода find.

##### Выполнение:

```
```
```

```
use learn
```

```
```
```



```
mongosh mongodb://127.0.0.1:27020/test>
test> use learn
switched to db learn
learn>
```

```
```
```

```
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'});
```

```
```
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

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('6835eb9d7a943e0cde6c4bd0') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6835eb9d7a943e0cde6c4bd1') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6835eb9d7a943e0cde6c4bd2') }
}
learn> db.unicorns.insert({name: 'Rooooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6835eb9e7a943e0cde6c4bd3') }
}
learn> db.unicorns.insert({name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6835eb9e7a943e0cde6c4bd4') }
}
learn> db.unicorns.insert({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
```

```
***
unicorn = {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender:
'm', vampires: 165};
db.unicorns.insert(unicorn);
***
```

```
learn> unicorn = {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(unicorn)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6835f4557a943e0cde6c4bdb') }
}
learn>
```

```
***
db.unicorns.find();
***
```

```

 mongosh mongodb://127.0.0.1:27017/?directConnection=true

learn> db.unicorns.find()
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd3'),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {

```

```

 mongosh mongodb://127.0.0.1:27017/?directConnection=true

vampires: 2
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6835ebb27a943e0cde6c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6835f4557a943e0cde6c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
learn>

```

### Задание 2.2.1

#### Описание:

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

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

#### Выполнение:

```

```
db.unicorns.find({gender: "m"});
```

```
db.unicorns.find({gender: "f"});
```

```
db.unicorns.find({gender: "f"}).limit(3);
```

```
db.unicorns.find({gender: "m"}).sort({name: 1});
```

```
db.unicorns.find({gender: "f"}).sort({name: 1});
```

```



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true

learn> db.unicorns.find({gender: "m"})
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd7'),
    name: 'Moo',
    loves: [ 'milk' ],
    weight: 1200,
    gender: 'm',
    vampires: 100
  }
]
```

```

 mongosh mongodb://127.0.0.1:27017/?directConnection=true

learn> db.unicorns.find({gender: "f"})
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {

```

```

 mongosh mongodb://127.0.0.1:27017/?directConnection=true&

learn>

learn>

learn>

learn>

learn> db.unicorns.find({gender: "f"}).limit(3)
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
]
learn>

```

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSe
learn> db.unicorns.find({gender: "m"}).sort({name: 1})
[
  {
    _id: ObjectId('6835f4557a943e0cde6c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {

```

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSe
learn> db.unicorns.find({gender: "f"}).sort({name: 1})
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('6835ebb27a943e0cde6c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd4'),

```

...

```
db.unicorns.find({gender: "f", loves: "carrot"});
```

```
db.unicorns.findOne({gender: "f", loves: "carrot"});
```



```
db.unicorns.find({gender: "f", loves: "carrot"}).limit(1);
...
```

```
learn> db.unicorns.find({gender: "f", loves: "carrot"})
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('6835ebb27a943e0cde6c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn>
```

```
learn> db.unicorns.findOne({gender: "f", loves: "carrot"})
{
  _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn> _
```

```
learn> db.unicorns.find({gender: "f", loves: "carrot"}).limit(1)
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

## Задание 2.2.2

### Описание:

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

### Выполнение:

```
...
db.unicorns.find({gender: "m"}, {loves: 0, gender: 0});
...
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTi
learn> db.unicorns.find({gender: "m"}, {likes: 0, gender: 0})
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd0'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd2'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd3'),
    name: 'Rooooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd6'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd7'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd9'),
    name: 'Pilot',

```

### Задание 2.2.3

#### Описание:

Вывести список единорогов в обратном порядке добавления.

#### Выполнение:

```
...
db.unicorns.find().sort({$natural: -1});
...
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true

learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('6835f4557a943e0cde6c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6835ebb27a943e0cde6c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd7'),

```

### Задание 2.2.4

#### Описание:

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

#### Выполнение:

```
...
db.unicorns.find({}, {_id: 0, loves: {$slice: 1}});
...
```

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelect
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: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    name: 'Solnara',
    loves: [ 'apple' ],
    weight: 550,
    gender: 'f',
  }
]

```

### Задание 2.3.1

#### Описание:

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

#### Выполнение:

```

...
db.unicorns.find(
  {
    gender: 'f',
    weight: { $gte: 500, $lte: 700 }
  },
  {
    _id: 0
  }
);
...

```

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true

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'
  }
]
learn>

```

### Задание 2.3.2

#### Описание:

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

#### Выполнение:

```

...
db.unicorns.find(
  {
    gender: 'm',
    weight: { $gte: 500 },
    loves: { $all: ['grape', 'lemon'] }
  },
  {
    _id: 0
  }
);
...

```

```
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
  }
]
learn> _
```

### Задание 2.3.3

#### Описание:

Найти всех единорогов, не имеющих ключ "vampires".

#### Выполнение:

```
...
db.unicorns.find(
  {
    vampires: { $exists: false }
  }
);
...
```

```
learn> db.unicorns.find(
...   {
...     vampires: { $exists: false }
...   }
... );
[
  {
    _id: ObjectId('6835ebb27a943e0cde6c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn>
```

### Задание 2.3.4

#### Описание:

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

## Выполнение:

```
...
db.unicorns.find(
  { gender: 'm' },
  {
    _id: 0,
    name: 1,
    loves: { $slice: 1 }
  }
).sort({ name: 1 });
...
```



```
learn> 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' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Rooooooodles', loves: [ 'apple' ] },
  { name: 'Unicrom', loves: [ 'energon' ] }
]
learn> _
```

### Задание 3.1.1

#### Описание:

1) Создайте коллекцию "towns", включающую следующие документы:

```
...
{
  name: "Punxsutawney ",
  population: 6200,
  last_sensus: ISODate("2008-01-31"),
  famous_for: [""],
  mayor: {
    name: "Jim Wehrle"
  }
}

{
  name: "New York",
  population: 22200000,
  last_sensus: ISODate("2009-07-31"),
  famous_for: ["statue of liberty", "food"],
  mayor: {
```

```

        name: "Michael Bloomberg",
        party: "I"
    }
}

{
    name: "Portland",
    population: 528000,
    last_sensus: ISODate("2009-07-20"),
    famous_for: ["beer", "food"],
    mayor: {
        name: "Sam Adams",
        party: "D"
    }
}
...

```

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

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

### Выполнение:

```

...

db.towns.insertMany([
    {
        name: "Punxsutawney",
        population: 6200,
        last_sensus: ISODate("2008-01-31"),
        famous_for: [""],
        mayor: {
            name: "Jim Wehrle"
        }
    },
    {
        name: "New York",
        population: 22200000,
        last_sensus: ISODate("2009-07-31"),
        famous_for: ["statue of liberty", "food"],
        mayor: {

```



```

        name: "Michael Bloomberg",
        party: "I"
    }
},
{
    name: "Portland",
    population: 528000,
    last_sensus: ISODate("2009-07-20"),
    famous_for: ["beer", "food"],
    mayor: {
        name: "Sam Adams",
        party: "D"
    }
}
]);
```

```

```

learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     population: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: [""],
...     mayor: {
...       name: "Jim Wehrle"
...     }
...   },
...   {
...     name: "New York",
...     population: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["statue of liberty", "food"],
...     mayor: {
...       name: "Michael Bloomberg",
...       party: "I"
...     }
...   },
...   {
...     name: "Portland",
...     population: 528000,
...     last_sensus: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: {
...       name: "Sam Adams",
...       party: "D"
...     }
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683619787a943e0cde6c4bdf'),
    '1': ObjectId('683619787a943e0cde6c4be0'),
    '2': ObjectId('683619787a943e0cde6c4be1')
  }
}
learn>

```

```

```

db.towns.find(
  { "mayor.party": "I" },
  {
    _id: 0,
    name: 1,

```

```

    mayor: 1
  }
);
```

```

```

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

```

```

```

db.towns.find(
  { "mayor.party": { $exists: false } },
  {
    _id: 0,
    name: 1,
    mayor: 1
  }
);
```

```

```

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

```

### Задание 3.1.2

#### Описание:

- 1) Сформировать функцию для вывода списка самцов единорогов.
- 2) Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.
- 3) Вывести результат, используя `forEach`.
- 4) Содержание коллекции 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', 44, 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'});
db.unicorns.insert ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704,
gender: 'm', vampires: 165});
...

```

## Выполнение:

```

...

fn = function() { return this.gender == "m"; };
db.unicorns.find(fn);
...

```

```

learn> fn = function() { return this.gender == "m"; };
[Function: fn]
learn> db.unicorns.find(fn);
MongoInvalidArgumentError: Query filter must be a plain object or ObjectId
learn>

```

(В новых версиях MongoDB не рекомендуется использовать функции в качестве фильтра для метода `find()`. Лучше использовать объект фильтра.)

```

...

filter = { gender: "m" };
db.unicorns.find(filter);
...

```

```

mongosh mongodb://127.0.0.1:27017/?directConnection=true
learn> filter = { gender: "m" };
{ gender: 'm' }
learn> db.unicorns.find(filter);
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd3'),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd9'),
    ...

```

```

...
var cursor = db.unicorns.find({ gender: "m" }).sort({ name: 1 }).limit(2);
cursor.forEach(function(unicorn) {
  printjson(unicorn);
});
...

```

```

learn> var cursor = db.unicorns.find({ gender: "m" }).sort({ name: 1 }).limit(2);
learn> cursor.forEach(function(unicorn) {
...   printjson(unicorn);
... });
{
  _id: ObjectId('6835f4557a943e0cde6c4bdb'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('6835eb9d7a943e0cde6c4bd0'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
learn>

```

### Задание 3.2.1

#### Описание:

Вывести количество самок единорогов весом от полутонны до 600 кг.

#### Выполнение:

```

...
db.unicorns.find({
  gender: 'f',
  weight: { $gte: 500, $lte: 600 }
}).count()
...

```

```

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

```

### Задание 3.2.2

#### Описание:

Вывести список предпочтений.

#### Выполнение:

```

...
db.unicorns.distinct("loves");
...

```

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

### Задание 3.2.3

#### Описание:

Посчитать количество особей единорогов обоих полов.

#### Выполнение:

```
...
db.unicorns.aggregate(
  {
    $group: {
      _id: "$gender",
      count: { $sum: 1 }
    }
  }
);
...
```

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

### Задание 3.3.1

#### Описание:

1) Выполнить команду:

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

2) Проверить содержимое коллекции "unicorns".

#### Выполнение:

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

```
db.unicorns.find();
...
```

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

(В современных версиях MongoDB метод `'save()'` считается устаревшим. Вместо него можно использовать, например, методы `'insertOne()'`, `'insertMany()'`, `'updateOne()'`, и `'updateMany()'`).

```
...
```

```
db.unicorns.insertOne({name: 'Barney', loves: ['grape'], weight: 340, gender:
'm'}));
db.unicorns.find();
...
```

```
learn> db.unicorns.insertOne({name: 'Barney', loves: ['grape'], weight: 340, gender: 'm'});
{
  acknowledged: true,
  insertedId: ObjectId('6837307a5a3d4a0ac76c4bd0')
}
learn> db.unicorns.find();
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
]
```

```

    },
    {
      _id: ObjectId('6835eb9e7a943e0cde6c4bd9'),
      name: 'Pilot',
      loves: [ 'apple', 'watermelon' ],
      weight: 650,
      gender: 'm',
      vampires: 54
    },
    {
      _id: ObjectId('6835ebb27a943e0cde6c4bda'),
      name: 'Nimue',
      loves: [ 'grape', 'carrot' ],
      weight: 540,
      gender: 'f'
    },
    {
      _id: ObjectId('6835f4557a943e0cde6c4bdb'),
      name: 'Dunx',
      loves: [ 'grape', 'watermelon' ],
      weight: 704,
      gender: 'm',
      vampires: 165
    },
    {
      _id: ObjectId('6837307a5a3d4a0ac76c4bd0'),
      name: 'Barney',
      loves: [ 'grape' ],
      weight: 340,
      gender: 'm'
    }
  ]
}
learn>

```

### Задание 3.3.2

#### Описание:

- 1) Для самки единорога "Ауна" внести изменения в БД: теперь ее вес 800, она убила 51 вампира.
- 2) Проверить содержимое коллекции "unicorns".

#### Выполнение:

```

...
db.unicorns.updateOne(
  { name: 'Ayna' },
  {
    $set: {
      weight: 800,
      vampires: 51
    }
  }
);
db.unicorns.find();
...

```



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

```
},
{
  _id: ObjectId('6835eb9e7a943e0cde6c4bd5'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 800,
  gender: 'f',
  vampires: 51
},
{
```

### Задание 3.3.3

#### Описание:

- 1) Для самца единорога "Raleigh" внести изменения в БД: теперь он любит "redbull".
- 2) Проверить содержимое коллекции "unicorns".

#### Выполнение:

```
...
db.unicorns.updateOne(
  { name: 'Raleigh' },
  {
    $set: {
      loves: ['redbull']
    }
  }
);
db.unicorns.find();
...
```

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

```
},
{
  _id: ObjectId('6835eb9e7a943e0cde6c4bd7'),
  name: 'Raleigh',
  loves: [ 'redbull' ],
  weight: 421,
  gender: 'm',
  vampires: 2
},
{
```

### Задание 3.3.4

#### Описание:

- 1) Всем самцам единорогов увеличить количество убитых вампиров на 5.
- 2) Проверить содержимое коллекции “unicorns”.

#### Выполнение:

```
...
db.unicorns.updateMany(
  { gender: 'm' },
  {
    $inc: { vampires: 5 }
  }
);
db.unicorns.find();
...
```

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

```
learn> db.unicorns.find();
[
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6835eb9d7a943e0cde6c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd3'),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('6835eb9e7a943e0cde6c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],

```

### Задание 3.3.5

#### Описание:

- 1) Изменить информацию о городе Портленд: мэр этого города теперь беспартийный.
- 2) Проверить содержимое коллекции "towns".

#### Выполнение:

```
...
db.towns.updateOne(
  { name: "Portland" },
  { $unset: { "mayor.party": 1 } }
);
db.towns.find()
...
```

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

```
learn> db.towns.find();
[
  {
    _id: ObjectId('683619787a943e0cde6c4bdf'),
    name: 'Punxsutawney',
    population: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('683619787a943e0cde6c4be0'),
    name: 'New York',
    population: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('683619787a943e0cde6c4be1'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
learn> _
```

### Задание 3.3.6

#### Описание:

- 1) Изменить информацию о самце единорога Pilot: теперь он любит и "chocolate".
- 2) Проверить содержимое коллекции "unicorns".

#### Выполнение:

```
...
db.unicorns.updateOne(
  { name: 'Pilot' },
  { $push: { loves: 'chocolate' } }
);
db.unicorns.find();
...
```

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

```
},
{
  _id: ObjectId('6835eb9e7a943e0cde6c4bd9'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon', 'chocolate' ],
  weight: 650,
  gender: 'm',
  vampires: 59
},
{
```

### Задание 3.3.7

#### Описание:

- 1) Изменить информацию о самке единорога Ауры: теперь она любит еще и "sugar", и "lemons".
- 2) Проверить содержимое коллекции "unicorns".

#### Выполнение:

```
...
db.unicorns.updateOne(
  { name: 'Aurora' },
  { $addToSet: { loves: { $each: ['sugar', 'lemons'] } } }
);
db.unicorns.find();
...
```

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

```
},
{
  _id: ObjectId('6835eb9d7a943e0cde6c4bd1'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
```

### Задание 3.4.1

#### Описание:

1) Создайте коллекцию "towns", включающую следующие документы:

```
...  
{  
  name: "Punxsutawney ",  
  population: 6200,  
  last_sensus: ISODate("2008-01-31"),  
  famous_for: ["phil the groundhog"],  
  mayor: {  
    name: "Jim Wehrle"  
  }  
}  
{  
  name: "New York",  
  population: 22200000,  
  last_sensus: ISODate("2009-07-31"),  
  famous_for: ["statue of liberty", "food"],  
  mayor: {  
    name: "Michael Bloomberg",  
    party: "I"  
  }  
}  
{  
  name: "Portland",  
  population: 528000,  
  last_sensus: ISODate("2009-07-20"),  
  famous_for: ["beer", "food"],  
  mayor: {  
    name: "Sam Adams",  
    party: "D"  
  }  
}  
...
```

2) Удалите документы с беспартийными мэрами.

3) Проверьте содержание коллекции.

4) Очистите коллекцию.

5) Просмотрите список доступных коллекций.

#### Выполнение:

```
...
```

```

db.towns.insertMany([
  {
    name: "Punxsutawney",
    population: 6200,
    last_sensus: ISODate("2008-01-31"),
    famous_for: ["phil the groundhog"],
    mayor: {
      name: "Jim Wehrle"
    }
  },
  {
    name: "New York",
    population: 22200000,
    last_sensus: ISODate("2009-07-31"),
    famous_for: ["statue of liberty", "food"],
    mayor: {
      name: "Michael Bloomberg",
      party: "I"
    }
  },
  {
    name: "Portland",
    population: 528000,
    last_sensus: ISODate("2009-07-20"),
    famous_for: ["beer", "food"],
    mayor: {
      name: "Sam Adams",
      party: "D"
    }
  }
]);

```

```

learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     population: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: ["phil the groundhog"],
...     mayor: {
...       name: "Jim Wehrle"
...     }
...   },
...   {
...     name: "New York",
...     population: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["statue of liberty", "food"],
...     mayor: {
...       name: "Michael Bloomberg",
...       party: "I"
...     }
...   },
...   {
...     name: "Portland",
...     population: 528000,
...     last_sensus: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: {
...       name: "Sam Adams",
...       party: "D"
...     }
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683746d85a3d4a0ac76c4bd1'),
    '1': ObjectId('683746d85a3d4a0ac76c4bd2'),
    '2': ObjectId('683746d85a3d4a0ac76c4bd3')
  }
}
learn>

```

(Метод `remove()` считается устаревшим, вместо него рекомендуется использовать `deleteMany()` или `deleteOne()`.)

```

...

```

```

db.towns.deleteMany({ "mayor.party": { $exists: false } });

```

```

db.towns.find();

```

```

...

```

```

learn> db.towns.deleteMany({ "mayor.party": { $exists: false } });
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find();
[
  {
    _id: ObjectId('683746d85a3d4a0ac76c4bd2'),
    name: 'New York',
    population: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('683746d85a3d4a0ac76c4bd3'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>

```

```

...

```

```

db.towns.deleteMany({});

```

```

show collections;

```



...

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

(Коллекция “towns” сохранилась. Чтобы полностью её удалить, можно использовать метод `drop()`.)

### **Задание 4.1.1**

#### **Описание:**

- 1) Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.
- 2) Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания.
- 3) Проверьте содержание коллекции единорогов.

#### **Выполнение:**

...

```
db.habitats.insertMany([  
  {  
    _id: "forest",  
    name: "Enchanted Forest",  
    description: "A magical forest where unicorns roam freely among ancient  
trees."  
  },  
  {  
    _id: "mountain",  
    name: "Crystal Mountains",  
    description: "High peaks with crystal caves, home to the most majestic  
unicorns."  
  },  
  {  
    _id: "meadow",  
    name: "Golden Meadows",  
    description: "Vast fields of golden grass where unicorns graze and  
play."  
  },  
  {
```

```

        _id: "volcano",
        name: "Volcanic Highlands",
        description: "Dangerous volcanic region inhabited by fire-resistant
unicorns."
    }
});
```

```

```

learn> db.habitats.insertMany([
...   {
...     _id: "forest",
...     name: "Enchanted Forest",
...     description: "A magical forest where unicorns roam freely among ancient trees."
...   },
...   {
...     _id: "mountain",
...     name: "Crystal Mountains",
...     description: "High peaks with crystal caves, home to the most majestic unicorns."
...   },
...   {
...     _id: "meadow",
...     name: "Golden Meadows",
...     description: "Vast fields of golden grass where unicorns graze and play."
...   },
...   {
...     _id: "volcano",
...     name: "Volcanic Highlands",
...     description: "Dangerous volcanic region inhabited by fire-resistant unicorns."
...   }
... ]);
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'mountain', '2': 'meadow', '3': 'volcano' }
}
learn>

```

```

```

```

```

db.unicorns.updateOne(
  { name: "Horny" },
  { $set: { habitat: { $ref: "habitats", $id: "forest" } } }
);

```

```

db.unicorns.updateOne(
  { name: "Aurora" },
  { $set: { habitat: { $ref: "habitats", $id: "meadow" } } }
);

```

```

db.unicorns.updateOne(
  { name: "Unicrom" },
  { $set: { habitat: { $ref: "habitats", $id: "volcano" } } }
);

```

```

db.unicorns.updateOne(
  { name: "Solnara" },
  { $set: { habitat: { $ref: "habitats", $id: "mountain" } } }
);

```

```

db.unicorns.updateOne(

```

```

    { name: "Dunx" },
    { $set: { habitat: { $ref: "habitats", $id: "volcano" } } }
  );

db.unicorns.find()
```

```

```

learn> db.unicorns.updateOne(
...   { name: "Horny" },
...   { $set: { habitat: { $ref: "habitats", $id: "forest" } } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne(
...   { name: "Aurora" },
...   { $set: { habitat: { $ref: "habitats", $id: "meadow" } } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne(
...   { name: "Unicrom" },
...   { $set: { habitat: { $ref: "habitats", $id: "volcano" } } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne(
...   { name: "Solnara" },
...   { $set: { habitat: { $ref: "habitats", $id: "mountain" } } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne(
...   { name: "Dunx" },

```

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('68374a085a3d4a0ac76c4bd4'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63,
    habitat: DBRef('habitats', 'forest')
  },
  {
    _id: ObjectId('68374a085a3d4a0ac76c4bd5'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef('habitats', 'meadow')
  },
  {
    _id: ObjectId('68374a085a3d4a0ac76c4bd6'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182,
    habitat: DBRef('habitats', 'volcano')
  },
  {
    _id: ObjectId('68374a085a3d4a0ac76c4bd7'),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('68374a085a3d4a0ac76c4bd8'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80,
    habitat: DBRef('habitats', 'mountain')
  },
]
```

### Задание 4.2.1

#### Описание:

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

2) Содержание коллекции единорогов "unicorns":

...

```
db.unicorns.insert({name: 'Horny', dob: new Date(1992,2,13,7,47), loves:
['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
db.unicorns.insert({name: 'Aurora', dob: new Date(1991, 0, 24, 13, 0), loves:
['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
db.unicorns.insert({name: 'Unicrom', dob: new Date(1973, 1, 9, 22, 10), loves:
['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
db.unicorns.insert({name: 'Rooooooodles', dob: new Date(1979, 7, 18, 18, 44),
loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
db.unicorns.insert({name: 'Solnara', dob: new Date(1985, 6, 4, 2, 1),
loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
```

```

db.unicorns.insert({name:'Ayna', dob: new Date(1998, 2, 7, 8, 30), loves:
['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
db.unicorns.insert({name:'Kenny', dob: new Date(1997, 6, 1, 10, 42), loves:
['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
db.unicorns.insert({name: 'Raleigh', dob: new Date(2005, 4, 3, 0, 57), loves:
['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
db.unicorns.insert({name: 'Leia', dob: new Date(2001, 9, 8, 14, 53), loves:
['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
db.unicorns.insert({name: 'Pilot', dob: new Date(1997, 2, 1, 5, 3), loves:
['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
db.unicorns.insert({name: 'Nimue', dob: new Date(1999, 11, 20, 16, 15), loves:
['grape', 'carrot'], weight: 540, gender: 'f'});
db.unicorns.insert({name: 'Dunx', dob: new Date(1976, 6, 18, 18, 18), loves:
['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165});
...

```

### Выполнение:

```

...
db.unicorns.createIndex({ name: 1 }, { unique: true });
...

```

```

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

```

(Подобный индекс удалось создать, так как все значения “name” в коллекции уникальны.)

### Задание 4.3.1

#### Описание:

- 1) Получите информацию о всех индексах коллекции "unicorns".
- 2) Удалите все индексы, кроме индекса для идентификатора.
- 3) Попытайтесь удалить индекс для идентификатора.

### Выполнение:

```

...
db.unicorns.getIndexes();
...

```

```

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

```

```

...
db.unicorns.dropIndex("name_1");
db.unicorns.getIndexes();
...

```

```

learn> db.unicorns.dropIndex("name_1");
{ nIndexesWas: 2, ok: 1 }
learn> db.unicorns.getIndexes();
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn>

```

```

...
db.unicorns.dropIndex("_id_");
...

```

```

learn> db.unicorns.dropIndex("_id_");
MongoServerError[InvalidOptions]: cannot drop _id index
learn> db.unicorns.getIndexes();
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn>

```

(Индекс для идентификатора нельзя удалить.)

#### **Задание 4.4.1**

##### **Описание:**

1) Создайте объемную коллекцию numbers, задействовав курсор:

```

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

```

2) Выберите последних четыре документа.

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

4) Создайте индекс для ключа "value".

5) Получите информацию о всех индексах коллекции "numbers".

6) Выполните запрос 2.

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

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

##### **Выполнение:**

```

...

```

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

```
learn> for(i = 0; i < 100000; i++){db.numbers.insert({value: i});}
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68375dfa5a3d4a0ac76dd28b') }
}
learn> db.numbers.count();
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
100000
learn> 
```

...

```
db.numbers.find().sort({value: -1}).limit(4);
db.numbers.find().sort({value: -1}).limit(4).explain("executionStats");
...
```

```
learn> db.numbers.find().sort({value: -1}).limit(4);
[
  { _id: ObjectId('68375dfa5a3d4a0ac76dd28b'), value: 99999 },
  { _id: ObjectId('68375dfa5a3d4a0ac76dd28a'), value: 99998 },
  { _id: ObjectId('68375dfa5a3d4a0ac76dd289'), value: 99997 },
  { _id: ObjectId('68375dfa5a3d4a0ac76dd288'), value: 99996 }
]
learn> db.numbers.find().sort({value: -1}).limit(4).explain("executionStats");
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'BA27D965',
    planCacheShapeHash: 'BA27D965',
    planCacheKey: '7A892B81',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'SORT',
      sortPattern: { value: -1 },
      memLimit: 104857600,
      limitAmount: 4,
      type: 'simple',
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 224,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      isCached: false,
      stage: 'SORT',
      nReturned: 4,
      executionTimeMillisEstimate: 118,
      works: 100006,
      advanced: 4,

```

(На выполнение запроса потребовалось 224 мс.)

...

```
db.numbers.createIndex({value: 1});
db.numbers.getIndexes();
...
```

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

...

```
db.numbers.find().sort({value: -1}).limit(4).explain("executionStats");
```

...

```
learn> db.numbers.find().sort({value: -1}).limit(4).explain("executionStats");
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'BA27D965',
    planCacheShapeHash: 'BA27D965',
    planCacheKey: '7A892B81',
    optimizationTimeMillis: 81,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: {
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { value: 1 },
          indexName: 'value_1',
          isMultiKey: false,
          multiKeyPaths: { value: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'backward',
          indexBounds: { value: [ '[MaxKey, MinKey]' ] }
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 81,
    totalKeysExamined: 4,
    totalDocsExamined: 4,
    executionStages: {
      isCached: false,
```

(На выполнение запроса потребовалось 81 мс. Данный запрос более эффективен, так как благодаря индексу “value\_1” удалось добиться ускорения выполнения запроса практически в три раза.)

### 3 Вывод

В рамках данной работы я получил опыт работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.



