

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: Залетов А.Д.

Факультет: ИКТ

Группа: K3239

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы	3
Практическое задание	3
Выполнение задания.....	4
Вывод.....	31

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.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 6.0.6 (текущая).

Выполнение работы

1) Создайте базу данных *Learn*.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Please enter a MongoDB connection string (Default: mongodb://localhost/): mongodb://localhost:27017/
mongodb://localhost:27017/
Current Mongosh Log ID: 65746aaa7f1b2361c84f4846
Connecting to:      mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.1
Using MongoDB:      7.0.4
Using Mongosh:       2.1.1

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

-----
The server generated these startup warnings when booting
2023-12-09T15:51:52.650+03:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

test> use learn
switched to db learn
learn> {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
...
...
...
...
... ;
Uncaught:
SyntaxError: Missing semicolon. (1:20)
```

2) Заполните коллекцию единорогов *unicorns*:

```
test> use learn
switched to db learn
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: { '_id': ObjectId('657460937a1450b0581efdbf6') }
}
learn> db.unicorns.insertOne({name: 'Horny', loves: ['carrot', 'papaya'], weight: 600, gender: 'm', vampires: 63});
{
  acknowledged: true,
  insertedId: ObjectId('65746094fa1450b0581efdbf7')
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('65746090a1450b0581efdbf8') }
}
learn> db.unicorns.insert({name: 'Unicorn', loves: ['amergon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('65746090a1450b0581efdbf9') }
}
learn> db.unicorns.insert({name: 'Nooonoodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('65746090a1450b0581efdbfa') }
}
learn> db.unicorns.insert({name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('65746090a1450b0581efdbfb') }
}
learn> db.unicorns.insert({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('65746090a1450b0581efdbfc') }
}
learn> db.unicorns.insert({name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('65746090a1450b0581efdbfd') }
}
learn> db.unicorns.insert({name: 'Kaleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('65746090a1450b0581efdbfe') }
}
learn> db.unicorns.insert({name: 'Lala', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('65746091a1450b0581efdbff') }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('65746091a1450b0581efd00') }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedId: { '_id': ObjectId('6574609c5a1450b0581efd001') }
}
```

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
...   gender: 'm',
...   vampires: 165
... })
TypeError: db.unicorns.save is not a function
learn> db.unicorns.save({
...   name: 'Dunx',
...   loves: ['grape', 'watermelon'],
...   weight: 704,
...   gender: 'm',
...   vampires: 165
... })
TypeError: db.unicorns.save is not a function
learn> db.unicorns.save({
...   name: 'Dunx',
...   loves: ['grape', 'watermelon'],
...   weight: 704,
...   gender: 'm',
...   vampires: 165
... });
TypeError: db.unicorns.save is not a function
learn> var lostunicorn=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})

learn> db.unicorns.insertone(lostunicorn);
TypeError: db.unicorns.insertone is not a function
learn> db.unicorns.insertOne(lostunicorn);
{
  acknowledged: true,
  insertedId: ObjectId('65746ce27f1b2361c84f4847')
}
```

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find()
[
  {
    _id: ObjectId('65746937a1456b6581efd6f6'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65746990a1456b6581efd6f8'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65746990a1456b6581efd6f9'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```



```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
{
  _id: ObjectId('65746991a1456b6581efd700'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon' ],
  weight: 650,
  gender: 'm',
  vampires: 54
},
{
  _id: ObjectId('657469c5a1456b6581efd701'),
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f'
},
{
  _id: ObjectId('65746ce27f1b2361c84f4847'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
]
learn> _
```

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.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({ gender: 'f' }).sort({ name: 1 }).limit(3).pretty()
[
  {
    _id: ObjectId('65746990a1456b6581efd6f8'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65746990a1456b6581efd6fc'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65746991a1456b6581efd6ff'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({ gender: 'm' }).sort({ name: 1 }).limit(3).pretty()
[
  {
    _id: ObjectId('65746ce27f1b2361c84f4847'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65746937a1456b6581efd6f6'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65746990a1456b6581efd6fd'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn>
  name: 'Aurora',
learn>
  weight: 450,
learn>
  vampires: 43
learn>

learn>

learn>

learn> db.unicorns.findOne({ gender: 'f', loves: 'carrot' })
{
  _id: ObjectId('65746990a1456b6581efd6f8'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```



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

{
  _id: ObjectId('65746990a1456b6581efd6fb'),
  name: 'Solnara',
  loves: [ 'apple', 'carrot', 'chocolate' ],
  weight: 550,
  gender: 'f',
  vampires: 80
},
{
  _id: ObjectId('657469c5a1456b6581efd701'),
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f'
}
]
learn> db.unicorns.find({ gender: 'f', loves: 'carrot' }).limit(1).pretty()
[
  {
    _id: ObjectId('65746990a1456b6581efd6f8'),
    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 предпочтениях и поле.

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

learn> db.unicorns.find({ gender: 'm' }, { loves: 0, gender: 0 }).sort({ name: 1 }).pretty()
[
  {
    _id: ObjectId('65746ce27f1b2361c84f4847'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('65746937a1456b6581efd6f6'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('65746990a1456b6581efd6fd'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('65746991a1456b6581efd700'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
]
```

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
[
  {
    _id: ObjectId('65746991a1456b6581efd700'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('65746991a1456b6581efd6fe'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('65746990a1456b6581efd6fa'),
    name: 'Rooooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('65746990a1456b6581efd6f9'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
]
learn> _
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find().sort({ _id: -1 }).pretty()
[
  {
    _id: ObjectId('65746ce27f1b2361c84f4847'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('657469c5a1456b6581efd701'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('65746991a1456b6581efd700'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
  },
  {
    _id: ObjectId('65746990a1456b6581efd6fa'),
    name: 'Rooooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('65746991a1456b6581efd6fe'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('65746990a1456b6581efd6f9'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
]
learn>
```

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn>
},
{
  _id: ObjectId('65746991a1456b6581efd6ff'),
  name: 'Leia',
  loves: [ 'apple', 'watermelon' ],
  weight: 601,
  gender: 'F',
  vampires: 33
},
{
  _id: ObjectId('65746991a1456b6581efd6fe'),
  name: 'Raleigh',
  loves: [ 'apple', 'sugar' ],
  weight: 421,
  gender: 'm',
  vampires: 2
},
{
  _id: ObjectId('65746990a1456b6581efd6fd'),
  name: 'Kenny',
  loves: [ 'grape', 'lemon' ],
  weight: 690,
  gender: 'm',
  vampires: 39
},
{
```

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
vampires: 39
},
{
  _id: ObjectId('65746990a1456b6581efd6fc'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 733,
  gender: 'F',
  vampires: 40
},
{
  _id: ObjectId('65746990a1456b6581efd6fb'),
  name: 'Solnara',
  loves: [ 'apple', 'carrot', 'chocolate' ],
  weight: 550,
  gender: 'F',
  vampires: 80
},
{
  _id: ObjectId('65746990a1456b6581efd6fa'),
  name: 'Rooooooodles',
  loves: [ 'apple' ],
  weight: 575,
  gender: 'm',
  vampires: 99
},
{
```

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
vampires: 99
},
{
  _id: ObjectId('65746990a1456b6581efd6f9'),
  name: 'Unicrom',
  loves: [ 'energon', 'redbull' ],
  weight: 984,
  gender: 'm',
  vampires: 182
},
{
  _id: ObjectId('65746990a1456b6581efd6f8'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
  _id: ObjectId('65746937a1456b6581efd6f6'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

Практическое задание 2.3.1:

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
{ name: 'Pilot', firstLove: 'apple' },
{ name: 'Nimue', firstLove: 'grape' },
{ name: 'Dunx', firstLove: 'grape' }
]
learn> db.unicorns.find({ gender: 'f', weight: { $gte: 500, $lte: 700 } }, { _id: 0 }).pretty()
[
  {
    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.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
vampires: 80
},
{
  name: 'Leia',
  loves: [ 'apple', 'watermelon' ],
  weight: 601,
  gender: 'f',
  vampires: 33
},
{
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f'
}
]
learn> db.unicorns.find({ gender: 'm', weight: { $gte: 500 }, loves: { $all: [ 'grape', 'lemon' ] } }, { _id: 0 }).pretty()
[
  {
    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 } }).pretty()
[
  {
    _id: ObjectId('657469c5a1456b6581efd701'),
    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.

```

mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
name: 'Kenny',
loves: [ 'grape', 'lemon' ],
weight: 690,
gender: 'm',
vampires: 39
}
]
learn> db.unicorns.find({ vampires: { $exists: false } }).pretty()
[
  {
    _id: ObjectId('657469c5a1456b6581efd701'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn> db.unicorns.find({ gender: 'm' }, { _id: 0, name: 1, firstLove: { $arrayElemAt: ['$loves', 0] } }).sort({ name: 1 }).pretty()
[
  { name: 'Dunx', firstLove: 'grape' },
  { name: 'Horny', firstLove: 'carrot' },
  { name: 'Kenny', firstLove: 'grape' },
  { name: 'Pilot', firstLove: 'apple' },
  { name: 'Raleigh', firstLove: 'apple' },
  { name: 'Roooooodles', firstLove: 'apple' },
  { name: 'Unicrom', firstLove: 'energon' }
]

```

Практическое задание 3.1.1:

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: ["status 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: ["status 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('657474a57f1b2301a4f84a0'),
  1: ObjectId('657474a57f1b2301a4f84a1'),
  2: ObjectId('657474a57f1b2301a4f84a2')
}
learn>

```

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn>
learn>
learn>
learn>
learn>
learn>
learn>
learn>
learn>
learn>
learn> db.towns.find({ "mayor.party": "I" }, { _id: 0, name: 1, "mayor.name": 1, "mayor.party": 1 }).pretty()
[
  {
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
learn> _
```

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn>
learn>
learn>
learn> db.towns.find({ "mayor.party": { $exists: false } }, { _id: 0, name: 1, "mayor.name": 1 }).pretty()
[ { name: 'Punxsutawney', mayor: { name: 'Jim Wehrle' } } ]
learn>
```

Практическое задание 3.1.2:

1) Сформировать функцию для вывода списка самцов единорогов.

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

learn>
learn>
learn>
learn> db.towns.find({ "mayor.party": { $exists: false } }, { _id: 0, name: 1, "mayor.name": 1 }).pretty()
[ { name: 'Punxsutawney', mayor: { name: 'Jim Wehrle' } } ]
learn>
learn>
learn> function listMaleUnicorns() {
...   return db.unicorns.find({ gender: 'm' }, { _id: 0, name: 1, weight: 1, loves: 1 }).pretty();
... }
[Function: listMaleUnicorns]
learn> listMaleUnicorns();
[
  { name: 'Horny', loves: [ 'carrot', 'papaya' ], weight: 600 },
  { name: 'Unicrom', loves: [ 'energon', 'redbull' ], weight: 984 },
  { name: 'Rooooooodles', loves: [ 'apple' ], weight: 575 },
  { name: 'Kenny', loves: [ 'grape', 'lemon' ], weight: 690 },
  { name: 'Raleigh', loves: [ 'apple', 'sugar' ], weight: 421 },
  { name: 'Pilot', loves: [ 'apple', 'watermelon' ], weight: 650 },
  { name: 'Dunx', loves: [ 'grape', 'watermelon' ], weight: 704 }
]
learn>
```

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

```
learn> var cursor = db.unicorns.find({ gender: 'm' }, { _id: 0, name: 1, weight: 1, loves: 1 }).sort({ name: 1 }).limit(2);
...   printjson(cursor.next());
... }
{
  name: 'Dunx', Id('657469c5a1456b6581efd701'),
  loves: [ 'Nimue',
    'grape', 'grape', 'carrot' ],
    'watermelon'
  ], gender: 'f'
  weight: 704
}
learn> db.unicorns.find({ gender: 'f', loves: 'carrot' }).limit(1).pretty()
name: 'Horny',
loves: [
  'carrot', ctId('65746990a1456b6581efd6f8'),
  'papaya urora',
], loves: [ 'carrot', 'grape' ],
weight: 60050,
gender: 'f',
```

3) Вывести результат, используя forEach.

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

learn>
learn>
learn>
learn> while (cursor.hasNext()) printjson(cursor.next());
learn>
learn> db.unicorns.find({ gender: 'm' }, { _id: 0, name: 1, weight: 1, loves: 1 }).sort({ name: 1 }).limit(2).forEach(doc => printjson(doc));
{
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704
}
{
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600
}
```


Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
learn> var femaleUnicornCount = db.unicorns.distinct("name", { gender: "f", weight: { $gte: 500, $lte: 600 } }).length;
learn> print("Количество самок: " + femaleUnicornCount);
Количество самок: 2
```

Практическое задание 3.2.2:

Вывести список предпочтений.

```
learn> db.unicorns.aggregate([{$unwind: "$loves"}, {$group: { _id: "$loves", count: { $sum: 1 } }}, {$project: { _id: 0, preference: "$_id", count: 1 } }]).pretty();
[ { count: 5, preference: 'apple' },
  { count: 1, preference: 'papaya' },
  { count: 4, preference: 'grape' },
  { count: 1, preference: 'energon' },
  { count: 1, preference: 'sugar' },
  { count: 1, preference: 'redbull' },
  { count: 3, preference: 'watermelon' },
  { count: 2, preference: 'lemon' },
  { count: 1, preference: 'strawberry' },
  { count: 4, preference: 'carrot' },
  { count: 1, preference: 'chocolate' }
```

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

```
learn> db.unicorns.aggregate([{$group: { _id: "$gender", count: { $sum: 1 } }}, {$project: { _id: 0, gender: "$_id", count: 1 } }]).pretty();
[ { count: 7, gender: 'm' }, { count: 5, gender: 'f' } ]
learn> _
```

Практическое задание 3.3.2:

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn>
learn>
learn>
learn>
learn>
learn> db.unicorns.updateOne({ name: 'Ayna', gender: 'f' }, { $set: { weight: 800, vampires: 51 } });
{ acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
learn> db.unicorns.find({ name: 'Ayna', gender: 'f' }).pretty();
[ { _id: ObjectId('65746990a1456b581efd6fc'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 800,
  gender: 'f',
  vampires: 51 } ]
```

Практическое задание 3.3.3:

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
insertedId: null,
matchedCount: 0,
modifiedCount: 1,
upsertedCount: 0
}
learn> db.unicorns.find({ name: 'Raleigh', gender: 'm' }).pretty();
[
  {
    _id: ObjectId('65746991a1456b6581efd6fe'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
learn> db.unicorns.updateMany({ gender: 'm' }, { $inc: { vampires: 5 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
learn> db.unicorns.find({ gender: 'm' }).pretty();
[
  {
    _id: ObjectId('65746991a1456b6581efd6fe'),
```

Практическое задание 3.3.4:

Всем самцам единорогов увеличить количество убитых вампиров на 5.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
}
learn> db.unicorns.updateMany({ gender: 'm' }, { $inc: { vampires: 5 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
learn> db.unicorns.find({ gender: 'm' }).pretty();
[
  {
    _id: ObjectId('65746937a1456b6581efd6f6'),
    name: 'Rorpy',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('65746990a1456b6581efd6f9'),
    name: 'Unicorn',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
]
```

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
},
{
  _id: ObjectId('65746990a1456b6581efd6fa'),
  name: 'Rooooooooodles',
  loves: [ 'apple' ],
  weight: 575,
  gender: 'm',
  vampires: 104
},
{
  _id: ObjectId('65746990a1456b6581efd6fd'),
  name: 'Fenny',
  loves: [ 'grape', 'lemon' ],
  weight: 690,
  gender: 'm',
  vampires: 44
},
{
  _id: ObjectId('65746991a1456b6581efd6fe'),
  name: 'Raleigh',
  loves: [ 'apple', 'sugar', 'redbull' ],
  weight: 421,
  gender: 'm',
  vampires: 7
},
{
  _id: ObjectId('65746991a1456b6581efd700'),
  name: 'Pilot',
```

```

mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
weight: 421,
gender: 'm',
vampires: 7
},
{
  _id: ObjectId('65746991a1456b6581efd700'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon' ],
  weight: 650,
  gender: 'm',
  vampires: 59
},
{
  _id: ObjectId('65746ce27f1b2361c84f4847'),
  name: 'Dunk',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 170
},
{
  _id: ObjectId('657479ae7f1b2361c84f484e'),
  name: 'Barry',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm',
  vampires: 5
}

```

Практическое задание 3.3.5:

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

Проверить содержимое коллекции towns.

```

mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
{
  _id: ObjectId('657479ae7f1b2361c84f484e'),
  name: 'Barry',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm',
  vampires: 5
}
learn> db.towns.updateOne({ name: 'Portland' }, { $set: { 'mayor.party': null } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({ name: 'Portland' }).pretty();
{
  _id: ObjectId('657474767f1b2361c84f484a'),
  name: 'Portland',
  population: 528000,
  last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
  famous_for: [ 'beer', 'food' ],
  mayor: { name: 'Sam Adams', party: null }
}

```

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' }, { $addToSet: { loves: 'chocolate' } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: 'Pilot', gender: 'm' }).pretty();
{
  _id: ObjectId('65746991a1456b6581efd700'),
  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' }, { $addToSet: { loves: { $each: ['sugar', 'lemon'] } } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: 'Aurora', gender: 'f' }).pretty();
{
  _id: ObjectId('65746990a1456b6581efd6f8'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Практическое задание 3.4.1:

Создайте коллекцию towns, включающую следующие документы:

```
{name: "Punxsutawney ",
  popujatiuon: 6200,
  last_sensus: ISODate("2008-01-31"),
  famous_for: ["phil the groundhog"],
  mayor: {
    name: "Jim Wehrle"
  }}

{name: "New York",
  popujatiuon: 22200000,
  last_sensus: ISODate("2009-07-31"),
  famous_for: ["status of liberty", "food"],
  mayor: {
    name: "Michael Bloomberg",
    party: "I"}}

{name: "Portland",
  popujatiuon: 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('65747bd17f1b2361c84f484f'),
    '1': ObjectId('65747bd17f1b2361c84f4850'),
    '2': ObjectId('65747bd17f1b2361c84f4851')
  }
}
```

2. Удалите документы с беспартийными мэрами.

```
learn> db.towns.insertMany([
... {name: "Punkoutaway", population: 6200, last_sensus: ISODate("2008-01-31"), famous_for: ["phil the groundhog"], mayor: {name: "Jim Wenhie"}},
... {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('65747bd17f1b2361c84f484f'),
    '1': ObjectId('65747bd17f1b2361c84f4850'),
    '2': ObjectId('65747bd17f1b2361c84f4851')
  }
}
learn> db.towns.deleteMany({"mayor.party": {$exists: false}});
```

3. Проверьте содержание коллекции.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65747bd17f1b2361c84f484f'),
    '1': ObjectId('65747bd17f1b2361c84f4850'),
    '2': ObjectId('65747bd17f1b2361c84f4851')
  }
}
learn> db.towns.deleteMany({"mayor.party": {$exists: false}});
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find().pretty();
[
  {
    _id: ObjectId('65747bd17f1b2361c84f4850'),
    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('65747bd17f1b2361c84f4851'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

4. Очистите коллекцию.

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

5. Просмотрите список доступных коллекций.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
'2': ObjectId('65747bd17f1b2361c84f4851')
}
}
learn> db.towns.deleteMany({"mayor.party": {$exists: false}});
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find().pretty();
[
  {
    _id: ObjectId('65747bd17f1b2361c84f4850'),
    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('65747bd17f1b2361c84f4851'),
    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({});
{ acknowledged: true, deletedCount: 2 }
learn> show collections;
towns
```

Практическое задание 4.1.1:

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
test>
test>
test>
test>
test> db.habitats.insertMany([
...   {
...     _id: 'forest',
...     name: 'Волшебный лес',
...     description: 'Зона обитания единорогов в лесу, где они находят уединение.'
...   },
...   {
...     _id: 'meadow',
...     name: 'Радужные луга',
...     description: 'Широкие луга, где единороги могут пастись.'
...   },
...   {
...     _id: 'mountain',
...     name: 'Вафельные горы',
...     description: 'Высокогорные зоны, где единороги могут обитать.'
...   },
... ]);
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'meadow', '2': 'mountain' }
```

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
insertedIds: { '0': 'forest', '1': 'meadow', '2': 'mountain' }
}
test> use habitats
switched to db habitats
habitats> db.unicorns.insertMany([
...   {
...     name: 'Horny',
...     habitat: { $ref: 'habitats', $id: 'forest' },
...     loves: ['carrot', 'papaya'],
...     weight: 600,
...     gender: 'm',
...     vampires: 63
...   },
...   {
...     name: 'Aurora',
...     habitat: { $ref: 'habitats', $id: 'meadow' },
...     loves: ['carrot', 'grape'],
...     weight: 450,
...     gender: 'f',
...     vampires: 43
...   },
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65772aa3d6a56d118be24b4c'),
    '1': ObjectId('65772aa3d6a56d118be24b4d')
  }
}
```

Проверьте содержание коллекции единорогов.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
insertedIds: {
  '0': ObjectId('65772aa3d6a56d118be24b4c'),
  '1': ObjectId('65772aa3d6a56d118be24b4d')
}
habitats> db.unicorns.find().pretty();
[
  {
    _id: ObjectId('65772aa3d6a56d118be24b4c'),
    name: 'Horny',
    habitat: DBRef('habitats', 'forest'),
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65772aa3d6a56d118be24b4d'),
    name: 'Aurora',
    habitat: DBRef('habitats', 'meadow'),
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
habitats>
```

Практическое задание 4.2.1:

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

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
objSlot: 11,
rootSlot: 9,
fieldBehavior: 'keep',
fields: [
]
unicorns> db.unicorns.find().pretty();db.unicorns.find().pretty();
],
unicorns>

unicorns> db.unicorns.find().pretty();

unicorns>

unicorns> db.unicorns.find().pretty();

unicorns> use learn
switched to db learn
learn> db.unicorns.createIndex({ name: 1 }, { unique: true });
name_1
learn> db.unicorns.find().pretty();
[
  {
    _id: ObjectId('65746937a1456b6581efd6f6'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('65746990a1456b6581efd6f8'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Содержание коллекции единорогов `unicorns`:

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

Практическое задание 4.3.1:

Получите информацию о всех индексах коллекции `unicorns`.

```
mongosh mongodb://localhost:27017/admin?connectTimeoutSeconds=200
vampires:44urned: 4,
{
  executionTimeMillisEstimate: 0,
  opens: 4,
  _id: ObjectId('65740991a145b06581ef06ff'),
  name: 'Raleigh',e: 0,
  loves: [ 'apple', 'sugar', 'redbull' ],
  weight: 4214f: 0,
  gender: 'm',reads: 4,
  vampires: 70rdSlot: 9,
  recordIdSlot: 10,
  seekKeySlot: 4,
  _id: ObjectId('65740991a145b06581ef06ff'),
  name: 'Ria',IdentSlot: 6,
  loves: [ 'apple', 'watermelon' ],
  weight: 601,xKeyPatternSlot: 8,
  gender: 'f',ds: [],
  vampires: 33utslots: []
},
{
  _id: ObjectId('65740991a145b06581ef0700'),
  name: 'Riot',
  loves: [ 'apple', 'watermelon', 'chocolate' ],
  weight: 640,
  gender: 'm',
  vampires: 59
},
{
  _id: ObjectId('65740991a145b06581ef0701'),
  name: 'Rise',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f'
},
{
  _id: ObjectId('65740ce27f1b2361c84f4847'),
  name: 'Sam',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 120
},
{
  _id: ObjectId('657479ae7f1b2361c84f484e'),
  name: 'Sunny',
  loves: [ 'grape' ],
  weight: 240,
  gender: 'm',
  vampires: 5
}
}
learn> db.unicorns.getIndexes();
{ v: 2, key: { _id: 1 }, name: '_id.' },
{ v: 2, key: { name: 1 }, name: 'name.1', unique: true }
learn>
```

Удалите все индексы, кроме индекса для идентификатора.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
{
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 170
},
{
  _id: ObjectId('657479ae7f1b2361c84f484e'),
  name: 'Barney',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm',
  vampires: 5
}
]
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> _
```

Попытайтесь удалить индекс для идентификатора.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
{
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 170
},
{
  _id: ObjectId('657479ae7f1b2361c84f484e'),
  name: 'Barney',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm',
  vampires: 5
}
]
learn> db.unicorns.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_1' },
  { 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> db.unicorns.dropIndex("_id_1");
MongoServerError: cannot drop _id index
learn>
```

Удалять индекс для идентификатора нельзя! Ибо это может сломать нашу БД, даже система не даёт нам этого сделать

Практическое задание 4.4.1:

Создайте объемную коллекцию `numbers`, задействовав курсор:

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

```
unicorns> for (let i = 0; i < 100000; i++) {  
...   db.numbers.insert({ value: i });  
... };  
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.  
  
;  
  
{  
  acknowledged: true,  
  insertedIds: { '0': ObjectId('65772ce0d6a56d118be3d1ed') }  
}
```

The screenshot shows the MongoDB Compass interface for the `unicorns.numbers` collection. The left sidebar displays a database structure with collections like `admin`, `config`, `habitats`, `learn`, `local`, `startup_log`, `task9db`, `test`, and `unicorns`. The `unicorns` collection is expanded, showing the `numbers` sub-collection. The main panel displays the `unicorns.numbers` collection with 100.0k documents and 1 index. The `Documents` tab is active, showing a list of documents with fields `_id` and `value`. The first five documents shown are:

_id	value
ObjectId('65772c7dd6a56d118be24b5c')	14
ObjectId('65772c7dd6a56d118be24b5d')	15
ObjectId('65772c7dd6a56d118be24b5e')	16
ObjectId('65772c7dd6a56d118be24b5f')	17
ObjectId('65772c7dd6a56d118be24b60')	18

Создайте индекс для ключа *value*.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
  closes: 1,
  saveState: 0,
  restoreState: 0,
  isEOF: 0,
  numReads: 4,
  recordSlot: 9,
  recordIdSlot: 10,
  seekKeySlot: 4,
  snapshotIdSlot: 5,
  indexIdSlot: 6,
  indexKeySlot: 7,
  indexKeyPatternSlot: 8,
  fields: [],
  outputSlots: []
}
}
}
}
}
unicorns>
value_1
unicorns>
unicorns>
unicorns>
unicorns>
unicorns>
unicorns> db.numbers.createIndex({ value: 1 });
```

Получите информацию о всех индексах коллекции *numbers*.

```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
  recordSlot: 9,
  recordIdSlot: 10,
  seekKeySlot: 4,
  snapshotIdSlot: 5,
  indexIdSlot: 6,
  indexKeySlot: 7,
  indexKeyPatternSlot: 8,
  fields: [],
  outputSlots: []
}
}
}
}
}
unicorns>
value_1
unicorns>
unicorns>
unicorns>
unicorns>
unicorns>
unicorns> db.numbers.createIndex({ value: 1 });
value_1
unicorns> db.numbers.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
```

Выполните запрос 2.

```

mongosh mongodb://localhost:27017/directConnections>true?serverSelectionTimeoutMS=3000
{
  "success": true,
  "executionTimeMillis": 12,
  "totalKeysExamined": 0,
  "totalDocsExamined": 0,
  "executionStages": [
    {
      "stage": "init",
      "planStage": 0,
      "returned": 0,
      "executionTimeMillisEstimate": 0,
      "open": 1,
      "close": 1,
      "saveState": 0,
      "restoreState": 0,
      "isRF": 0,
      "limit": 0,
      "inputStages": [
        {
          "stage": "none",
          "planStage": 0,
          "returned": 0,
          "executionTimeMillisEstimate": 0,
          "open": 1,
          "close": 1,
          "saveState": 0,
          "restoreState": 0,
          "isRF": 0,
          "objColl": 1,
          "rootColl": 0,
          "fieldBehavior": "keep",
          "fields": [
            {
              "projectFields": [],
              "projectIndex": 1,
              "forNameObject": true,
              "returnNameObject": false,
              "inputStages": [
                {
                  "stage": "none",
                  "planStage": 0,
                  "returned": 0,
                  "executionTimeMillisEstimate": 0,
                  "open": 1,
                  "close": 1,
                  "saveState": 0,
                  "restoreState": 0,
                  "isRF": 0,
                  "totalDocsExamined": 0,
                  "totalKeysExamined": 0,
                  "collectionScanned": 0,
                  "collectionScanned": 0,
                  "indexScanned": 0,
                  "indexScanned": 1,
                  "indexUsed": [
                    {
                      "innerOpen": 0,
                      "innerClose": 0,
                      "outerProject": 1,
                      "outerCorrelated": [
                        {
                          "long": 0,
                          "long": 0,
                          "long": 0,
                          "long": 0,
                          "long": 0
                        ],
                        {
                          "outerStage": {
                            "stage": "none",

```

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

В обоих случаях потребовалось 12 миллисекунд, скорее всего это произошло из-за того что выборка очень мала.

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

Запрос с индексами в MongoDB должен работать быстрее, чем без них, так как Индекс позволяет MongoDB эффективно находить необходимые документы, сокращая количество документов, которые нужно просмотреть компьютеру.

Вывод

В ходе лабораторной работы были освоены практические навыки по созданию, функций, документов и коллекций в СУБД MongoDB. Были созданы функции на выборку данных, а также были созданы запросы для удаления, замены и вставки данных разными способами. Также был создан курсор и различные запросы на выборку данных.