МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

**ФАКУЛЬТЕТ ПРИКЛАДНАЯ ИНФОРМАТИКА**

# ЛАБОРАТОРНАЯ РАБОТА № 6

по дисциплине

‘**Проектирование и реализация баз данных**’

*Выполнил:*
Мохаджер Алиреза
Студент группы K3240

*Преподаватель:*
Говорова Марина Михайловна

Санкт-Петербург, 2025

# 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.

# Выполнение

Практическое задание 2.1.1:

1)Создайте базу данных learn.
2)Заполните коллекцию единорогов unicorns:

```
test> use learn
switched to db learn
learn> db.createCollection('unicorns')
{ ok: 1 }
learn> db.unicorns.insertMany([
... {name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63},
... {name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43}
... ,{name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182},
... {name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99},
... {name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80},
... {name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40},
... {name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39},
... {name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2},
... {name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33},
... {name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54},
... {name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'}
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('684e3cfc0d899acff350eb67'),
    '1': ObjectId('684e3cfc0d899acff350eb68'),
    '2': ObjectId('684e3cfc0d899acff350eb69'),
    '3': ObjectId('684e3cfc0d899acff350eb6a'),
    '4': ObjectId('684e3cfc0d899acff350eb6b'),
    '5': ObjectId('684e3cfc0d899acff350eb6c'),
    '6': ObjectId('684e3cfc0d899acff350eb6d'),
    '7': ObjectId('684e3cfc0d899acff350eb6e'),
    '8': ObjectId('684e3cfc0d899acff350eb6f'),
    '9': ObjectId('684e3cfc0d899acff350eb70'),
    '10': ObjectId('684e3cfc0d899acff350eb71')
  }
}
```

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

```
learn> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document)
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('684e3de10d899acff350eb72') }
}
```

4) Проверьте содержимое коллекции с помощью метода find.

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb67'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb68'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb69'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6a'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6b'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6c'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6d'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6f'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb70'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb71'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('684e3de10d899acff350eb72'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
```

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.

Список самцов:

```
learn> db.unicorns.find({gender:"m"}).sort({name:1})
[
  {
    _id: ObjectId('684e3de10d899acff350eb72'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb67'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6d'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb70'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6a'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb69'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

Список первых трёх самок

```
learn> db.unicorns.find({gender:"f"}).sort({name:1}).limit(3)
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb68'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6c'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6f'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

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

findOne:

```
learn> db.unicorns.findOne({gender: "f", loves:"carrot"})
{
  _id: ObjectId('684e3cfc0d899acff350eb68'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Find and Limit:

```
learn> db.unicorns.find({gender: "f", loves:"carrot"}).limit(1)
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb68'),
    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.

```
learn> db.unicorns.find({gender: "m"}, {loves: 0, gender: 0}).sort({name: 1})
[
  {
    _id: ObjectId('684e3de10d899acff350eb72'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb67'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6d'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb70'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6e'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6a'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb69'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
]
```

Практическое задание 2.2.3:
Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('684e3de10d899acff350eb72'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb71'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb70'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6f'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6d'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6c'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6b'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6a'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb69'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb68'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb67'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
]
```

Практическое задание 2.1.4
Вывести список единорогов с названием первого любимого предпочтения, исключив идентификатор.

```
learn> db.unicorns.find({}, {loves:{$slice:1}, _id:0})
[
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Aurora',
    loves: [ 'carrot' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    name: 'Unicrom',
    loves: [ 'energon' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    name: 'Solnara',
    loves: [ 'apple' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    name: 'Ayna',
    loves: [ 'strawberry' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    name: 'Kenny',
    loves: [ 'grape' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    name: 'Raleigh',
    loves: [ 'apple' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    name: 'Leia',
    loves: [ 'apple' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    name: 'Pilot',
    loves: [ 'apple' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  { name: 'Nimue', loves: [ 'grape' ], weight: 540, gender: 'f' },
  {
    name: 'Dunx',
    loves: [ 'grape' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
```

Практическое задание 2.3.1
Вывести список самок единорогов весом от полутонны до 700 кг, исключив вывод идентификатора

```
learn> db.unicorns.find({gender:"f", weight:{$gte: 500 , $lte: 700}}, {_id:0})
[
  {
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.2
Вывести список самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора.

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

Практическое задание 2.3.3
Найти всех единорогов, не имеющих ключ vampires.

```
learn> db.unicorns.find({vampires: {$exists:false}})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb71'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.4
Вывести список упорядоченный список имен самцов единорогов с информацией об их первом предпочтении.

```
learn> db.unicorns.find({gender: "m"},{loves:{$slice:1}}).sort({name:1})
[
  {
    _id: ObjectId('684e3de10d899acff350eb72'),
    name: 'Dunx',
    loves: [ 'grape' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb67'),
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6d'),
    name: 'Kenny',
    loves: [ 'grape' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb70'),
    name: 'Pilot',
    loves: [ 'apple' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6e'),
    name: 'Raleigh',
    loves: [ 'apple' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb6a'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb69'),
    name: 'Unicrom',
    loves: [ 'energon' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

Практическое задание 3.1.1
1) Создайте коллекцию towns, включающую следующие документы
{name: "Punxsutawney ",
populatiuon: 6200,
last_sensus: ISODate("2008-01-31"), famous_for: [""],
mayor: {
name: "Jim Wehrle"
}}
{name: "New York",
populatiuon: 22200000,
last_sensus: ISODate("2009-07-31"), famous_for: ["status of liberty", "food"], mayor: {
name: "Michael Bloomberg", party: "I"}},
{name: "Portland",
populatiuon: 528000,
last_sensus: ISODate("2009-07-20"), famous_for: ["beer", "food"], mayor: {
name: "Sam Adams", party: "D"}}

```
learn> db.createCollection("towns")
{ ok: 1 }
```

```
learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     population: 6200,
...     last_census: ISODate("2008-01-31"),
...     famous_for: [""],
...     mayor: {
...       name: "Jim Wehrle"
...     }
...   },
...   {
...     name: "New York",
...     population: 22200000,
...     last_census: ISODate("2009-07-31"),
...     famous_for: ["statue of liberty", "food"],
...     mayor: {
...       name: "Michael Bloomberg",
...       party: "I"
...     }
...   },
...   {
...     name: "Portland",
...     population: 528000,
...     last_census: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: {
...       name: "Sam Adams",
...       party: "D"
...     }
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('684e85f30d899acff350eb73'),
    '1': ObjectId('684e85f30d899acff350eb74'),
    '2': ObjectId('684e85f30d899acff350eb75')
  }
}
```

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

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

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

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

Практическое задание 3.1.2
3) Сформировать функцию для вывода списка самцов единорогов.
4) Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.
5) Вывести результат, используя forEach.

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

Практическое задание 3.2.1
Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 3.2.2
Вывести список предпочтений.

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

Практическое задание 3.2.3
Посчитать количество особей единорогов обоих полов

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

Практическое задание 3.3.1
1) Выполнить команду:
> db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})

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

2) Проверить содержимое коллекции unicorns.

Практическое задание 3.3.2
Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.

```
learn> db.unicorns.find({name:"Ayna"})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb6c'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
]
learn> db.unicorns.updateOne({name:"Ayna"}, {$set:{weight:800 , vampires:51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:"Ayna"})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb6c'),
    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 рэдбул.

```
learn> db.unicorns.find({name:"Raleigh"})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb6e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
learn> db.unicorns.updateOne({name:"Raleigh"}, {$set:{loves:"redbull"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:"Raleigh"})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb6e'),
    name: 'Raleigh',
    loves: 'redbull',
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Практическое задание 3.3.4
Всем самцам единорогов увеличить количество убитых вапмиров на 5.
До увеличения:

```
learn> db.unicorns.find({gender:"m"})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb67'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb69'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
```

```
learn> db.unicorns.update({gender:"m"}, {$inc:{vampires:5}}, {multi: true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
```

После увеличения:

```
learn> db.unicorns.find({gender:"m"})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb67'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('684e3cfc0d899acff350eb69'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
```

Практическое задание 3.3.5
Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.

```
learn> db.towns.update({name:"Portland"}, {$unset:{"mayor.party":1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({name:"Portland"})
[
  {
    _id: ObjectId('684e85f30d899acff350eb75'),
    name: 'Portland',
    population: 528000,
    last_census: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

Практическое задание 3.3.6
Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.

```
learn> db.unicorns.find({name:"Pilot"})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb70'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
learn> db.unicorns.updateOne({name:"Pilot"}, {$push:{loves:"chocolate"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:"Pilot"})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb70'),
    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.

```
learn> db.unicorns.update({name: "Aurora"}, {$addToSet: {loves:{$each:["sugar", "lemon"]}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:"Aurora"})
[
  {
    _id: ObjectId('684e3cfc0d899acff350eb68'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 3.4.1

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"}}*

2. Удалите документы с беспартийными мэрами.
3. Проверьте содержание коллекции.
4. Очистите коллекцию.
5. Просмотрите список доступных коллекций.

```
learn> db.towns.find()
[
  {
    _id: ObjectId('684e85f30d899acff350eb73'),
    name: 'Punxsutawney',
    population: 6200,
    last_census: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('684e85f30d899acff350eb74'),
    name: 'New York',
    population: 22200000,
    last_census: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('684e85f30d899acff350eb75'),
    name: 'Portland',
    population: 528000,
    last_census: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

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

```
learn> db.towns.find()
[
  {
    _id: ObjectId('684e85f30d899acff350eb73'),
    name: 'Punxsutawney',
    population: 6200,
    last_census: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('684e85f30d899acff350eb75'),
    name: 'Portland',
    population: 528000,
    last_census: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

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

Практическое задание 4.1.1

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 едиорогов.

```
learn> db.createCollection("habitats")
{ ok: 1 }
learn> db.habitats.insertMany([{
... _id: "forest",fullName: "Волшебный лес",
... description: "Густой лес с древними деревьями, где единороги прячутся среди зарослей"},
... {
... _id: "mountains",fullName: "Хрустальные горы",
... description: "Высокие горы с кристально чистыми озерами, любимое место молодых единорогов"},
... {
... _id: "meadow",fullName: "Радужная поляна",
... description: "Просторная поляна, где растут волшебные цветы всех цветов радуги"}
... ])
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'mountains', '2': 'meadow' }
}
```

```
learn> db.unicorns.update(
... {gender: "f"},
... {$set: {habitat: {$ref: "habitats",$id: "meadow"}}},{multi: true}
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 5,
  modifiedCount: 5,
  upsertedCount: 0
}
learn> db.unicorns.update( { gender: "m" }, { $set: { habitat: { $ref: "habitats", $id: "mountains" } } }, { multi: true } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
```

```
learn> function getUnicornWithHabitat(unicornName) {
... let unicorn = db.unicorns.findOne({name: unicornName})
... if (unicorn && unicorn.habitat) {
... unicorn.habitatData = db[unicorn.habitat.$ref].findOne({_id: unicorn.habitat.$id})
... }
... return unicorn}
```

```
learn> getUnicornWithHabitat("Aurora")
{
  _id: ObjectId('684e3cfc0d899acff350eb68'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
  weight: 450,
  gender: 'f',
  vampires: 43,
  habitat: DBRef('habitats', 'meadow'),
  habitatData: null
}
```

Практическое задание 4.2.1
Проверьте, можно ли задать для коллекции unicorns индекс для ключа name с флагом unique.

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

Практическое задание 4.3.1

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

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

```
learn> db.unicorns.dropIndexes()
{
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
```

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

Практическое задание 4.4.1
1. Создайте объемную коллекцию numbers, задействовав курсор:

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

2. Выберите последних четыре документа.
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 numbres.
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?

Создание коллекции numbers

```
learn> for (i = 0; i < 100000; i++) { db.numbers.insert({ value: i })
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('684ec82e0d899acff3538745') }
}
```

Выбор последних четырех документов

```
learn> db.numbers.find().sort({_id: -1}).limit(4)
[
  { _id: ObjectId('684ec82e0d899acff3538745'), value: 99999 },
  { _id: ObjectId('684ec82e0d899acff3538744'), value: 99998 },
  { _id: ObjectId('684ec82e0d899acff3538743'), value: 99997 },
  { _id: ObjectId('684ec82e0d899acff3538742'), value: 99996 }
]
learn> db.numbers.find().sort({value: -1}).limit(4)
[
  { _id: ObjectId('684ec82e0d899acff3538745'), value: 99999 },
  { _id: ObjectId('684ec82e0d899acff3538744'), value: 99998 },
  { _id: ObjectId('684ec82e0d899acff3538743'), value: 99997 },
  { _id: ObjectId('684ec82e0d899acff3538742'), value: 99996 }
]
```

Анализ плана выполнения запроса

```
learn> var explain = db.numbers.find().sort({value: -1}).limit(4).explain("executionStats")

learn> explain.executionStats.executionTimeMillis
82
learn> explain
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'BA27D965',
    planCacheShapeHash: 'BA27D965',
    planCacheKey: '7A892B81',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
```

Создание индекса для ключа value

```
learn> db.numbers.ensureIndex({"value": 1}, {"unique": true})
[ 'value_1' ]
```

Получение информации об индексах

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

Повторное выполнение запроса 2

```
learn> db.numbers.find().sort({value: -1}).limit(4)
[
  { _id: ObjectId('684ec82e0d899acff3538745'), value: 99999 },
  { _id: ObjectId('684ec82e0d899acff3538744'), value: 99998 },
  { _id: ObjectId('684ec82e0d899acff3538743'), value: 99997 },
  { _id: ObjectId('684ec82e0d899acff3538742'), value: 99996 }
]
```

Анализ плана выполнения с индексом

```
learn> var explainWithIndex = db.numbers.find().sort({value: -1}).limit(4).explain("executionStats")

learn> explainWithIndex.executionStats.executionTimeMillis
0
learn> explainWithIndex
{
  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,
```

Сравнение времени выполнения

Без индекса: executionTimeMillis будет больше(у меня получилось 82ms)
С индексом: executionTimeMillis будет очень меньше (у меня получилось 0ms)

Вывод

В ходе лабораторной работы были успешно освоены ключевые аспекты работы с MongoDB, включая выполнение CRUD-операций, управление индексами, обработку сложных структур данных и оптимизацию запросов, что подтвердило эффективность этой СУБД для работы с документоориентированными данными.