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 Христофоров Владислав Николаевич
Факультет прикладной информатики
Группа K3240
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

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.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 8.0.9 (последняя)

2 CRUD-ОПЕРАЦИИ В СУБД MONGODB. ВСТАВКА ДАННЫХ. ВЫБОРКА ДАННЫХ

2.1 ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ

Практическое задание 2.1.1:

1) *Создайте базу данных learn.*

```
use learn
```

```
test> use learn
switched to db 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'});
```

```
learn> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63})
learn>
  acknowledged: true,
  insertedIds: { '0': ObjectId('68379ee4d5c0118ec76c4bdc') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68379eeed5c0118ec76c4bdd') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68379ef5d5c0118ec76c4bde') }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68379efcd5c0118ec76c4bdf') }
}
learn> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68379f02d5c0118ec76c4be0') }
}
learn> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68379f0cd5c0118ec76c4be1') }
}
learn> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68379f11d5c0118ec76c4be2') }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68379f1dd5c0118ec76c4be3') }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33})
{
  acknowledged: true,
learn>
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54})
learn>

  insertedIds: { '0': ObjectId('68379f2ad5c0118ec76c4be5') }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68379f2fd5c0118ec76c4be6') }
}
```

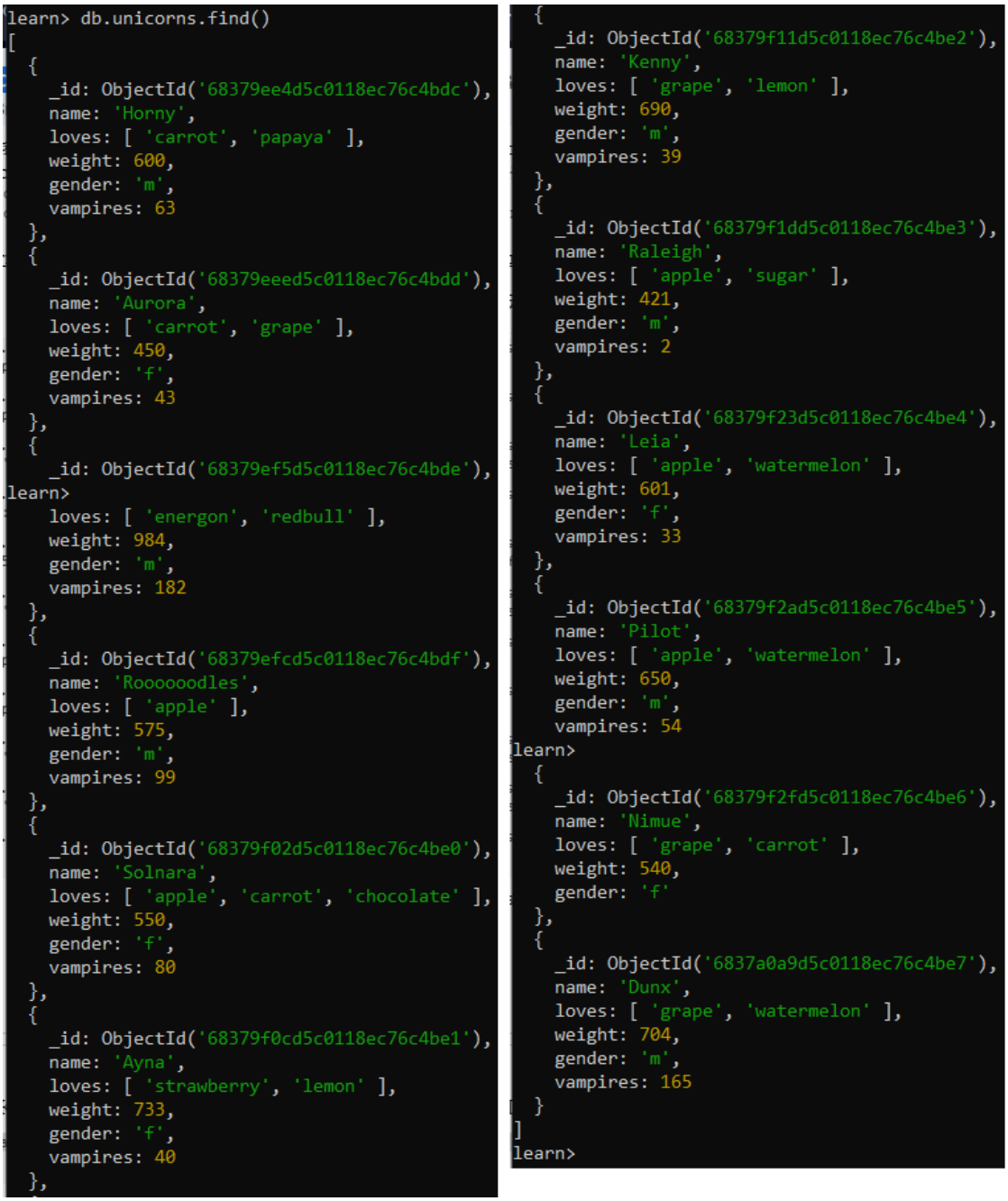
3) Используя второй способ, вставьте в коллекцию единорогов документ:

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

```
learn> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
... )
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6837a0a9d5c0118ec76c4be7') }
}
```

4) Проверьте содержимое коллекции с помощью метода find.

`dp.inicorns.find()`



```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68379eeed5c0118ec76c4bdd'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('68379ef5d5c0118ec76c4bde'),
learn>
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('68379efcd5c0118ec76c4bdf'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('68379f02d5c0118ec76c4be0'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('68379f0cd5c0118ec76c4be1'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('68379f11d5c0118ec76c4be2'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('68379f1dd5c0118ec76c4be3'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('68379f23d5c0118ec76c4be4'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('68379f2ad5c0118ec76c4be5'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
learn>
  {
    _id: ObjectId('68379f2fd5c0118ec76c4be6'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6837a0a9d5c0118ec76c4be7'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
learn>
```

2.2 ВЫБОРКА ДАННЫХ ИЗ БД

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.

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

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('6837a0a9d5c0118ec76c4be7'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68379f11d5c0118ec76c4be2'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('68379f2ad5c0118ec76c4be5'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('68379f1dd5c0118ec76c4be3'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('68379efcd5c0118ec76c4bdf'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('68379ef5d5c0118ec76c4bde'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

```
db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
```

```
learn> db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
[
  {
    _id: ObjectId('68379eeed5c0118ec76c4bdd'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('68379f0cd5c0118ec76c4be1'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('68379f23d5c0118ec76c4be4'),
    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.

```
db.unicorns.findOne({loves: 'carrot'})
```

```
learn> db.unicorns.findOne({loves: 'carrot'})
{
  _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

```
db.unicorns.find({loves: 'carrot'}).limit(1)
```

```
learn> db.unicorns.find({loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
]
```

Практическое задание 2.2.2:

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

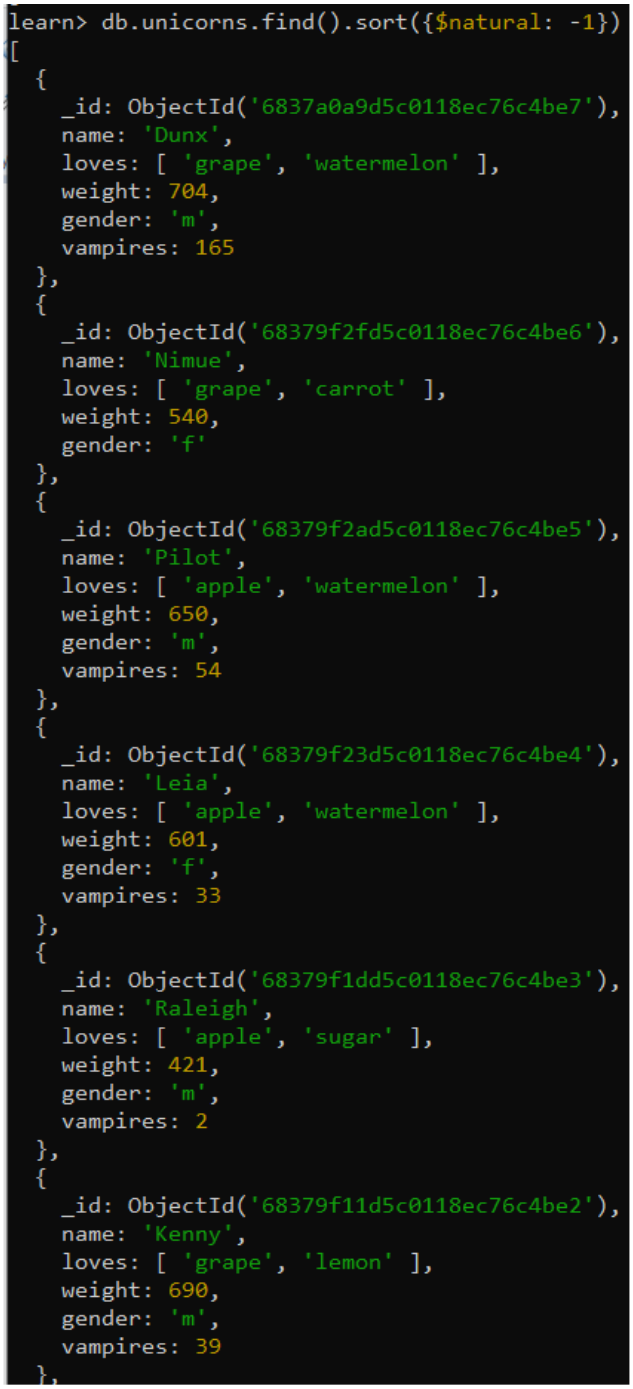
```
db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
```

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
[
  {
    _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('68379ef5d5c0118ec76c4bde'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId('68379efcd5c0118ec76c4bdf'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('68379f11d5c0118ec76c4be2'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('68379f1dd5c0118ec76c4be3'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('68379f2ad5c0118ec76c4be5'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('6837a0a9d5c0118ec76c4be7'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
]
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

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

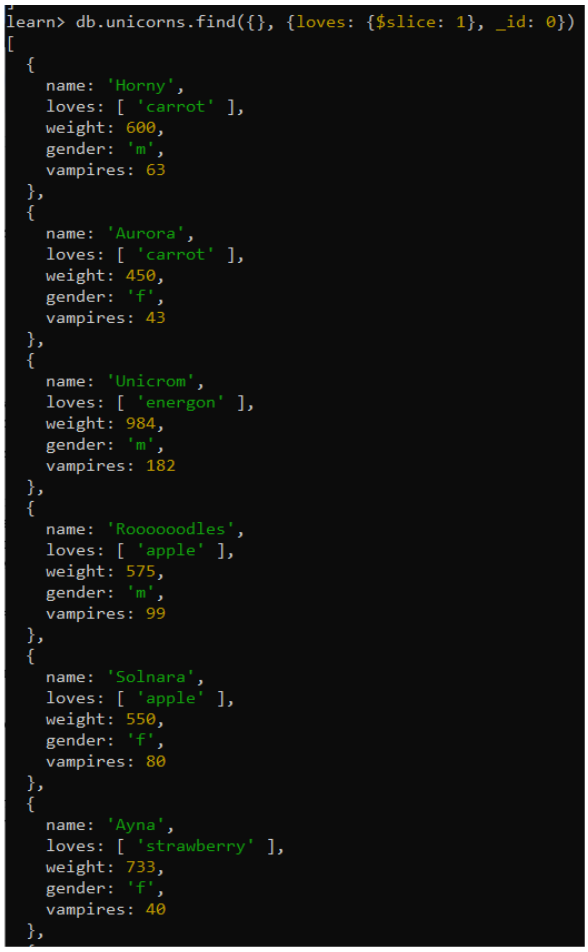


```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('6837a0a9d5c0118ec76c4be7'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('68379f2fd5c0118ec76c4be6'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('68379f2ad5c0118ec76c4be5'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('68379f23d5c0118ec76c4be4'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('68379f1dd5c0118ec76c4be3'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('68379f11d5c0118ec76c4be2'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('68379f0cd5c0118ec76c4be1'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('68379f02d5c0118ec76c4be0'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('68379efcd5c0118ec76c4bdf'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('68379ef5d5c0118ec76c4bde'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('68379eeed5c0118ec76c4bdd'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
]
```

Практическое задание 2.2.4:

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

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



```
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
  }
]
```

2.3 ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Практическое задание 2.3.1:

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

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

```
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.

```
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
  }
]
```

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

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

```
learn> db.unicorns.find({vampires: {$exists:false}})
[
  {
    _id: ObjectId('68379f2fd5c0118ec76c4be6'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.4:

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

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

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

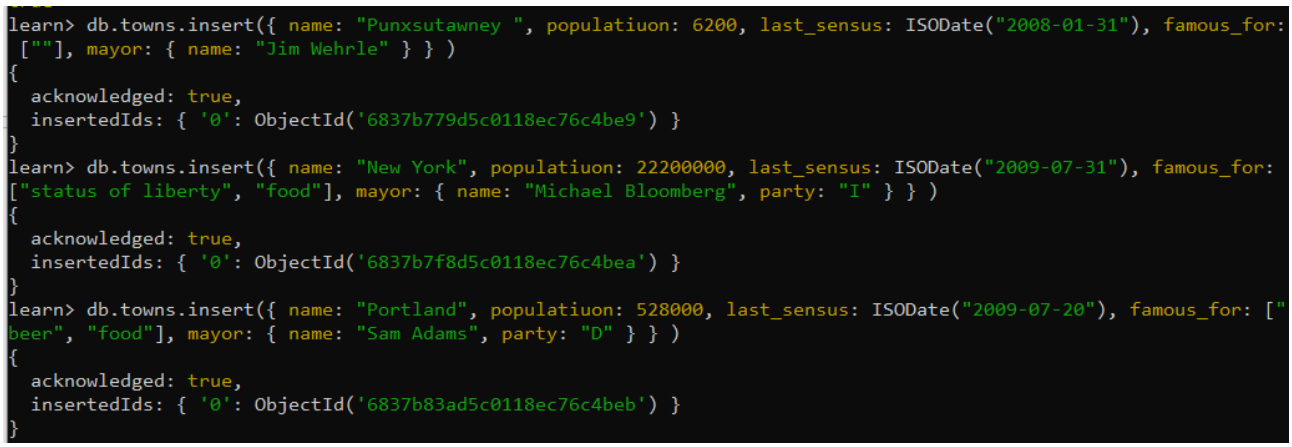
3 ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB. ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ

3.1 ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ

Практическое задание 3.1.1:

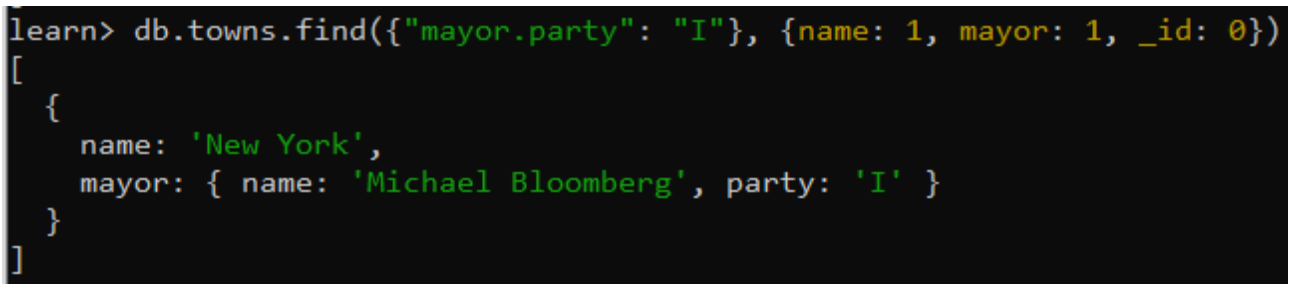
1) Создайте коллекцию towns

```
db.towns.insert({ name: "Punxsutawney ", populatiuon: 6200, last_sensus: ISODate("2008-01-31"), famous_for: [""], mayor: { name: "Jim Wehrle" } } )
db.towns.insert({ name: "Portland", populatiuon: 528000, last_sensus: ISODate("2009-07-20"), famous_for: ["beer", "food"], mayor: { name: "Sam Adams", party: "D" } } )
db.towns.insert({ name: "Portland", populatiuon: 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.

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



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

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

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

Практическое задание 3.1.2:

1) Сформировать функцию для вывода списка самцов единорогов.

```
findMaleUnicorns = function() { return this.gender == 'm'; }
db.unicorns.find({ $where: findMaleUnicorns } )
```

```
learn> findMaleUnicorns = function() { return this.gender == 'm'; }
[Function: findMaleUnicorns]
learn> db.unicorns.find({ $where: findMaleUnicorns } )
[
  {
    _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68379ef5d5c0118ec76c4bde'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('68379efcd5c0118ec76c4bdf'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('68379f11d5c0118ec76c4be2'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
```

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

```
var cursor = db.unicorns.find({ $where: findMaleUnicorns } );null;
cursor.sort({name:1}).limit(2);null;
```

```
learn> var cursor = db.unicorns.find({ $where: findMaleUnicorns } );null;
null
learn> cursor.sort({name:1}).limit(2);null;
null
```

3) *Вывести результат, используя forEach*

```
cursor.forEach(function(obj) { print(obj); })
```

```
learn> cursor.forEach(function(obj) { print(obj); })
{
  _id: ObjectId('6837a0a9d5c0118ec76c4be7'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

3.2 АГРЕГИРОВАННЫЕ ЗАПРОСЫ

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

```
learn> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 600}})
[
  {
    _id: ObjectId('68379f02d5c0118ec76c4be0'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('68379f2fd5c0118ec76c4be6'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 600}}).count()
2
```

Практическое задание 3.2.2:

Вывести список предпочтений.

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

```
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.

```
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 } ]
```

3.3 РЕДАКТИРОВАНИЕ ДАННЫХ

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
```

Пояснение: метод save() считается устаревшим в MongoDB

Практическое задание 3.3.2:

1) *Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.*

```
db.unicorns.update({name: 'Ayna', gender: 'f'}, {name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 800, gender: 'f', vampires: 51})
```

```
learn> db.unicorns.update({name: 'Ayna', gender: 'f'}, {name: 'Ayna', loves: ['strawberry', 'lemon'],
weight: 800, gender: 'f', vampires: 51})
MongoInvalidArgumentError: Update document requires atomic operators
```

Пояснение: метод update() считается устаревшим в MongoDB. Ошибка говорит о том, что нужно указать атомарный оператор.

```
db.unicorns.updateOne({name: 'Ayna', gender: 'f'}, {$set: {weight: 800, vampires: 51}})
```

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

2) *Проверить содержимое коллекции unicorns.*

```
learn> db.unicorns.find({name: 'Ayna', gender: 'f'})
[
  {
    _id: ObjectId('68379f0cd5c0118ec76c4be1'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

Практическое задание 3.3.3:

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

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

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

2) *Проверить содержимое коллекции unicorns.*

```
learn> db.unicorns.find({name: 'Raleigh', gender: 'm'})
[
  {
    _id: ObjectId('68379f1dd5c0118ec76c4be3'),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Практическое задание 3.3.4:

1) *Всем самцам единорогов увеличить количество убитых вапмиров на 5.*

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

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

2) *Проверить содержимое коллекции unicorns.*

```
learn> db.unicorns.find({gender: 'm'})
[
  {
    _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('68379ef5d5c0118ec76c4bde'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId('68379efcd5c0118ec76c4bdf'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104
  },
  {
    _id: ObjectId('68379f11d5c0118ec76c4be2'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 44
  },
  {
    _id: ObjectId('68379f1dd5c0118ec76c4be3'),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 7
  },
  {
    _id: ObjectId('68379f2ad5c0118ec76c4be5'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  },
  {
    _id: ObjectId('6837a0a9d5c0118ec76c4be7'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 170
  }
]
```

Практическое задание 3.3.5:

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

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

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

2) *Проверить содержимое коллекции towns*

```
learn> db.towns.find({name: 'Portland'})
[
  {
    _id: ObjectId('6837b83ad5c0118ec76c4beb'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

Практическое задание 3.3.6:

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

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

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

2) *Проверить содержимое коллекции unicorns*

```
learn> db.unicorns.find({name: 'Pilot', gender: 'm'})
[
  {
    _id: ObjectId('68379f2ad5c0118ec76c4be5'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

Практическое задание 3.3.7:

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

`db.unicorns.updateOne({name: 'Aurora', gender: 'f'}, {$addToSet: {loves: {$each: ['sugar', 'lemon']}}})`

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

2) *Проверить содержимое коллекции unicorns*

```
learn> db.unicorns.find({name: 'Aurora', gender: 'f'})
[
  {
    _id: ObjectId('68379eeed5c0118ec76c4bdd'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

3.4 УДАЛЕНИЕ ДАННЫХ ИЗ КОЛЛЕКЦИИ

Практическое задание 3.4.1:

1) *Удалите документы с беспартийными мэрами*

```
db.towns.remove({"mayor.party": {$exists: false}}))
```

```
learn> db.towns.remove({"mayor.party": {$exists: false}})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 2 }
```

2) *Проверьте содержимое коллекции*

```
learn> db.towns.find()
[
  {
    _id: ObjectId('6837b7f8d5c0118ec76c4bea'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
```

3) *Очистите коллекцию.*

```
learn> db.towns.remove({})
{ acknowledged: true, deletedCount: 1 }
```

4) *Просмотрите список доступных коллекций.*

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

4 ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

4.1 ССЫЛКИ В БД

Практическое задание 4.1.1:

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

```
db.habitats.insert({_id: 'rainbow', name: 'Rainbow Republic', desc: 'Unicorn Habitat'})
```

```
learn> db.habitats.insert({_id: 'rainbow', name: 'Rainbow Republic', desc: 'Unicorn Habitat'})
{ acknowledged: true, insertedIds: { '0': 'rainbow' } }
learn> db.habitats.find()
[
  { _id: 'rainbow', name: 'Rainbow Republic', desc: 'Unicorn Habitat' }
]
```

2) Включите для нескольких единорогов в документы ссылку на зону обитания, использую второй способ автоматического связывания

```
db.unicorns.updateMany({name: {$in: ['Horny', 'Aurora']}}, {$set: {habitat: {$ref: 'habitats', $id: 'rainbow'}}})
```

```
learn> db.unicorns.updateMany({name: {$in: ['Horny', 'Aurora']}}, {$set: {habitat: {$ref: 'habitats', $id: 'rainbow'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

3) Проверьте содержание коллекции едиорогов.

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('68379ee4d5c0118ec76c4bdc'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    habitat: DBRef('habitats', 'rainbow')
  },
  {
    _id: ObjectId('68379eeed5c0118ec76c4bdd'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef('habitats', 'rainbow')
  },
  {
```

4.2 НАСТРОЙКА ИНДЕКСОВ

Практическое задание 4.2.1:

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

```
db.unicorns.ensureIndex({'name': 1}, {'unique': true})
```

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

```
learn> db.unicorns.insert({name: 'Aurora'})
Uncaught:
MongoBulkWriteError: E11000 duplicate key error collection: learn.unicorns index: name_1 dup key: { name: "Aurora" }
Result: BulkWriteResult {
  insertedCount: 0,
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {},
  insertedIds: {}
}
Write Errors: [
  WriteError {
    err: {
      index: 0,
      code: 11000,
      errmsg: 'E11000 duplicate key error collection: learn.unicorns index: name_1 dup key: { name: "Aurora" }',
      errInfo: undefined,
      op: { name: 'Aurora', _id: ObjectId('68383d2fd5c0118ec76c4bec') }
    }
  }
]
```

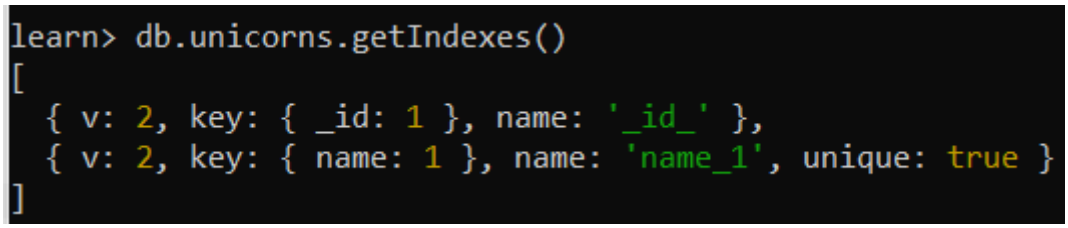
Ответ: да, в исходную коллекцию unicorns можно задать индекс для ключа name с флагом unique.

4.3 УПРАВЛЕНИЕ ИНДЕКСАМИ

Практическое задание 4.3.1:

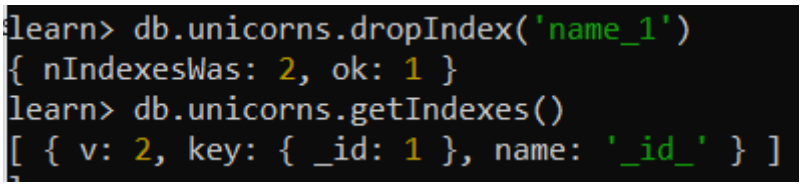
1) *Получите информацию о всех индексах коллекции* unicorns .

```
db.unicorns.getIndexes()
```



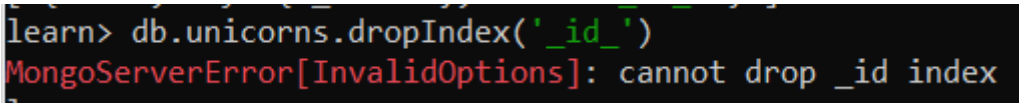
2) *Удалите все индексы, кроме индекса для идентификатора.*

```
db.unicorns.dropIndex('name_1')
```



3) *Попытайтесь удалить индекс для идентификатора.*

```
db.unicorns.dropIndex('_id_')
```



Вывод: удалить индекс для идентификатора нельзя, т. к. он необходим для идентификации документов и предотвращает вставку двух документов с одинаковым идентификатором.

4.4 ПЛАН ЗАПРОСА

Практическое задание 4.4.1:

1) *Создайте объемную коллекцию numbers, задействовав курсор:*

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

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68384129d5c0118ec76dd28c') }
}
```

2) *Выберите последних четыре документа.*

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

```
learn> db.numbers.find().sort({value: -1}).limit(4)
[
  { _id: ObjectId('68384129d5c0118ec76dd28c'), value: 99999 },
  { _id: ObjectId('68384129d5c0118ec76dd28b'), value: 99998 },
  { _id: ObjectId('68384129d5c0118ec76dd28a'), value: 99997 },
  { _id: ObjectId('68384129d5c0118ec76dd289'), value: 99996 }
]
```

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

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

```
winningPlan: {
  isCached: false,
  stage: 'SORT',
  sortPattern: { value: -1 },
  memLimit: 104857600,
  limitAmount: 4,
  type: 'simple',
  inputStage: { stage: 'COLLSCAN', direction: 'forward' }
},
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 136,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
  executionStages: {
    isCached: false,
    stage: 'SORT',
```

Ответ: на выполнение запроса 2 без индекса потребовалось 136 мс.

4) *Создайте индекс для ключа value.*

```
db.numbers.ensureIndex({'value': 1})
```

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

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

```
db.numbers.getIndexes()
```

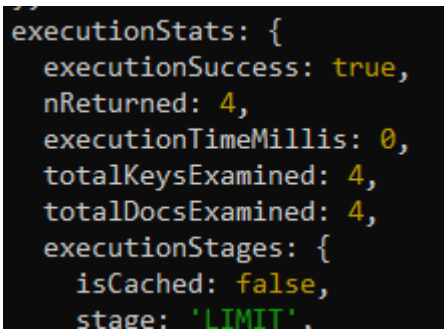
```
learn> db.numbers.getIndexes()
[   internalQueryFrameworkControl: 'trySbeRestricted',
  { v: 2, key: { _id: 1 }, name: '_id_' },lationForRegex: 1
  { v: 2, key: { value: 1 }, name: 'value_1' }
] ok: 1
```

6) *Выполните запрос 2.*

```
learn> db.numbers.find().sort({value: -1}).limit(4)
[
  { _id: ObjectId('68384129d5c0118ec76dd28c'), value: 99999 },
  { _id: ObjectId('68384129d5c0118ec76dd28b'), value: 99998 },
  { _id: ObjectId('68384129d5c0118ec76dd28a'), value: 99997 },
  { _id: ObjectId('68384129d5c0118ec76dd289'), value: 99996 }
]
```

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

```
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]' ] }
    }
  }
},
```



Ответ: на выполнение запроса 2 с индексом потребовалось 0 мс.

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

Ответ. Без индекса время выполнения запроса заметно больше, чем с индексом. Это связано с тем, что без индекса MongoDB вынужден делать полное сканирование коллекции (COLLSCAN), и только после этого сортировать все документы, чтобы отобрать 4 последних. С индексом же время почти нулевое, вместо COLLSCAN + SORT выполняется только IXSCAN, в котором вместо полного сканирования коллекции, происходит поиск документов с использованием индексов.

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы выполнены основные операции CRUD в MongoDB: вставка и выборка документов, использование логических операторов, обновление и удаление данных. Для более сложных задач применялись встроенные JavaScript-функции и курсоры, а также агрегатные запросы и работа с вложенными структурами. Также, показано влияние индексов на производительность выборки – без индекса запрос на последние 4 записи требовал значительного времени, тогда как наличие индекса по полю value сократило время практически до нуля.