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: Кахикало К.Р.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.

Санкт-Петербург 2024

Оглавление

Практическое задание:.....3
Схема базы данных:.....5
Ход работы:.....6

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).

Ход работы:

Практическое задание 2.1.1:

1) *Создайте базу данных learn.*
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: { '0': ObjectId('65d271308c0c47b7926d2116') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271308c0c47b7926d2117') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271308c0c47b7926d2118') }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271308c0c47b7926d2119') }
}
learn> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80}
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271318c0c47b7926d211a') }
}
learn> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271318c0c47b7926d211b') }
}
learn> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271318c0c47b7926d211c') }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271318c0c47b7926d211d') }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271318c0c47b7926d211e') }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271318c0c47b7926d211f') }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271318c0c47b7926d2120') }
}
```

Используя второй способ, вставьте в коллекцию единорогов документ:

```
learn> document = {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document);
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d271788c0c47b7926d2121') }
}
```

Проверьте содержимое коллекции с помощью метода find.

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('65d271308c0c47b7926d2116'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2117'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2118'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2119'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211a'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211b'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211c'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211d'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211d'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211e'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211f'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65d271318c0c47b7926d2120'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('65d271788c0c47b7926d2121'),
    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.

db.unicorns.find({gender: 'm'}).sort("name").limit(3)

```
learn> db.unicorns.find({gender: 'm'}).sort("name").limit(3)
[
  {
    _id: ObjectId('65d271788c0c47b7926d2121'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2116'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211c'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

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

```
learn> db.unicorns.find({gender: 'f'}).sort("name").limit(3)
[
  {
    _id: ObjectId('65d271308c0c47b7926d2117'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211b'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211e'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

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

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

```
learn> db.unicorns.findOne({gender: 'f', loves: "carrot"})
{
  _id: ObjectId('65d271308c0c47b7926d2117'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Практическое задание 2.2.2:

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

db.unicorns.find({gender: 'm'}, {gender:false, loves:false}).sort("name").limit(3)

```
learn> db.unicorns.find({gender: 'm'}, {gender:false, loves:false}).sort("name").limit(3)
[
  {
    _id: ObjectId('65d271788c0c47b7926d2121'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2116'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211c'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
]
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

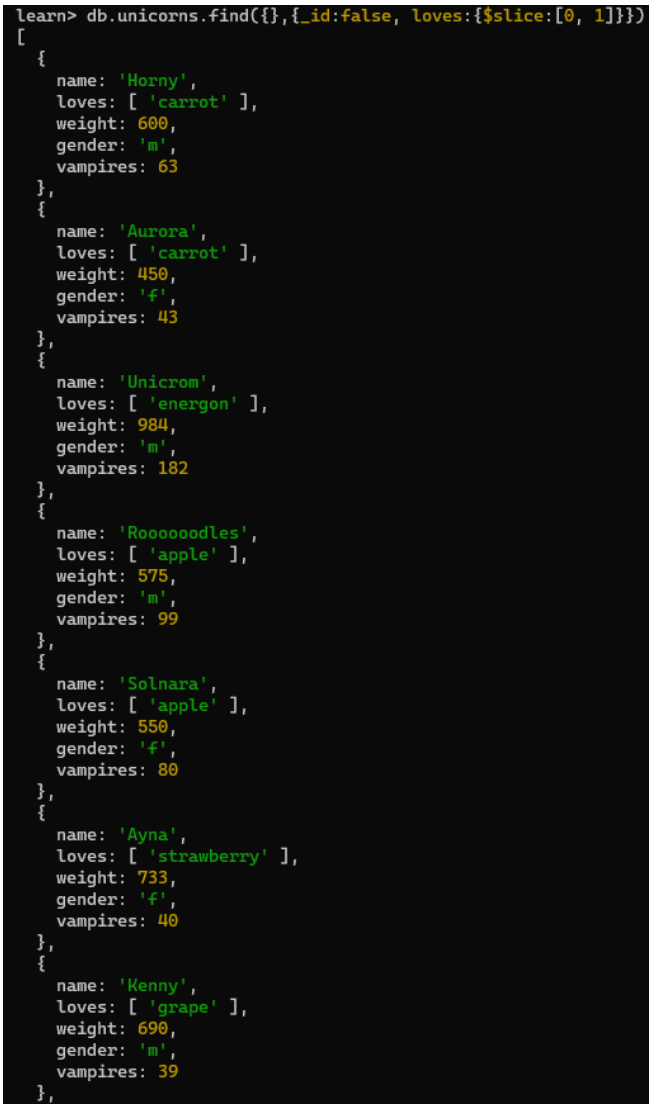
db.unicorns.find().sort({ $natural: -1 })

```
learn> db.unicorns.find().sort({ $natural: -1 })
[
  {
    _id: ObjectId('65d271788c0c47b7926d2121'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65d271318c0c47b7926d2120'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211f'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211e'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211d'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211c'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211b'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211a'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2119'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2118'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2117'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2116'),
    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.

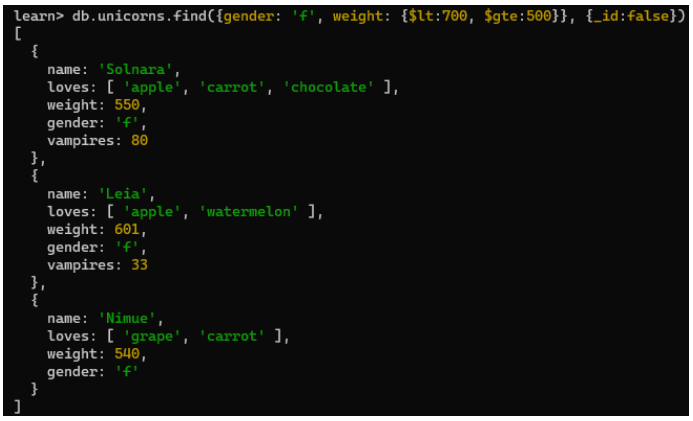
db.unicorns.find({},{_id:false, loves:{$slice:[0, 1]}})



Практическое задание 2.3.1:

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

db.unicorns.find({gender: 'f', weight: {$lt:700, $gte:500}}, {_id:false})



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:false})

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

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ `vampires`.

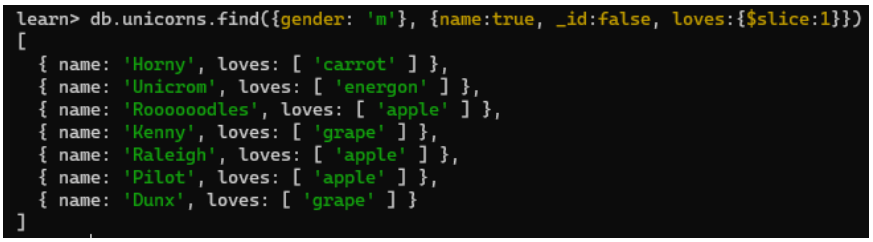
db.unicorns.find({vampires:{$exists:false}})

```
learn> db.unicorns.find({vampires:{$exists:false}})
[
  {
    _id: ObjectId('65d271318c0c47b7926d2120'),
    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.

db.unicorns.find({gender: 'm'}, {name:true, _id:false, loves:{$slice:1}})



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"}}
```

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

```
learn> db.towns.insert( { name: "Punxsutawney ", populatiuon: 6200, last_sensus: ISODate("2008-01-31"), famous_for: [""], mayor: { name: "Jim Wehrle" } } )
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d27dba8c0c47b7926d2122') }
}
learn> db.towns.insert({name: "New York",
... populatiuon: 22200000,
... last_sensus: ISODate("2009-07-31"),
... famous_for: ["status of liberty", "food"],
... mayor: {
...    name: "Michael Bloomberg",
... party: "I"}}
... )
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d27dce8c0c47b7926d2123') }
}
learn> db.towns.insert(
... {name: "Portland",
... populatiuon: 528000,
... last_sensus: ISODate("2009-07-20"),
... famous_for: ["beer", "food"],
... mayor: {
...    name: "Sam Adams",
... party: "D"}}
... )
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65d27de08c0c47b7926d2124') }
}
```

db.towns.find({"mayor.party":"I"}, {_id:false, name:true, mayor:true})

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

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

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

Практическое задание 3.1.2:

1) *Сформировать функцию для вывода списка самцов единорогов.*
2) *Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.*
3) *Вывести результат, используя forEach.*
4) *Содержание коллекции единорогов* `unicorns`*:*

```
db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});

db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});

db.unicorns.insert({name: 'Roooooodles', 44), loves: ['apple'], weight: 575, gender: 'm', vampires: 99});

db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});

db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});

db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});

db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});

db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});

db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});

db.unicorns.insert ({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});

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



find_males = function() { return db.unicorns.find({gender:'m'}); }

let cursor = find_males().sort("name").limit(2);

cursor.forEach(print);
```

```
learn> let cursor = find_males().sort("name").limit(2);

learn> cursor.forEach(print);
{
  _id: ObjectId('65d271788c0c47b7926d2121'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('65d271308c0c47b7926d2116'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

```
learn> db.unicorns.find({gender:"f", weight:{"$lt":600, $gte:500}}).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: 'f', count: 5 }, { _id: 'm', count: 7 } ]
```

Практическое задание 3.3.1:

1. *Выполнить команду:*

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

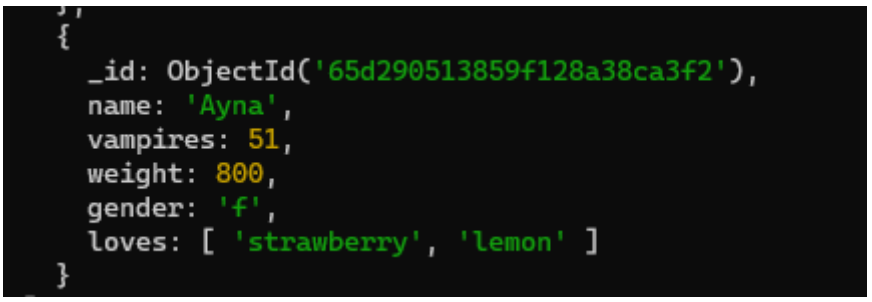
2. *Проверить содержимое коллекции* `unicorns`.

```
db.unicorns.findAndModify({query:{ name: 'Barny', loves: ['grape'], weight: 340, gender: 'm' }, upsert:true, update:{ name: 'Barny', loves: ['grape'], weight: 340, gender: 'm' }})
```

Практическое задание 3.3.2:

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

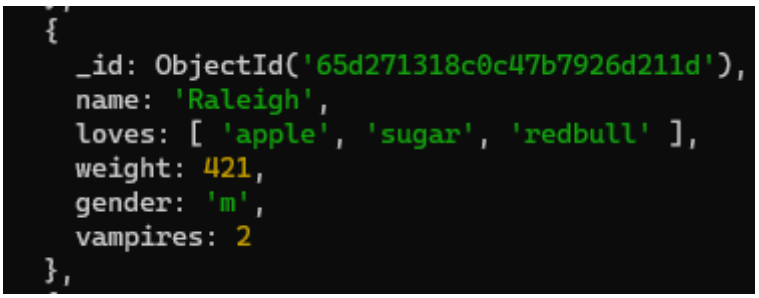
```
db.unicorns.updateOne({name:"Ayna"}, {$set:{weight:800, vampires:51}}, {upsert:true})
```



Практическое задание 3.3.3:

1. *Для самца единорога* `Raleigh` *внести изменения в БД: теперь он любит рэдбул.*
2. *Проверить содержимое коллекции* `unicorns`.

```
db.unicorns.updateOne({name:"Raleigh"}, {$addToSet:{loves:"redbull"}}, {upsert:true})
```



Практическое задание 3.3.4:

1. *Всем самцам единорогов увеличить количество убитых вапмиров на 5.*
2. *Проверить содержимое коллекции* `unicorns`.

```
db.unicorns.update({gender:"m"}, {$inc:{vampires:-5}}, {multi:true})
```

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('65d271308c0c47b7926d2116'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 58
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2117'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2118'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 177
  },
  {
    _id: ObjectId('65d271308c0c47b7926d2119'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 94
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211a'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211b'),
    weight: 800,
    vampires: 51
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211c'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 34
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211d'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: -3
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211e'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211f'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 49
  },
```

```
  {
    _id: ObjectId('65d271318c0c47b7926d211b'),
    weight: 800,
    vampires: 51
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211c'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 34
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211d'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: -3
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211e'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('65d271318c0c47b7926d211f'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 49
  },
  {
    _id: ObjectId('65d271318c0c47b7926d2120'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('65d271788c0c47b7926d2121'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 160
  },
  {
    _id: ObjectId('65d28e793859f128a38ca326'),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm',
    vampires: -5
  },
  {
    _id: ObjectId('65d28f8f3859f128a38ca39f'),
    weight: 800,
    vampires: 51
  },
  {
    _id: ObjectId('65d290513859f128a38ca3f2'),
    name: 'Ayna',
    vampires: 51,
    weight: 800,
    gender: 'f',
    loves: [ 'strawberry', 'lemon' ]
  }
```

Практическое задание 3.3.5:

1. *Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.*
2. *Проверить содержимое коллекции* `towns`.

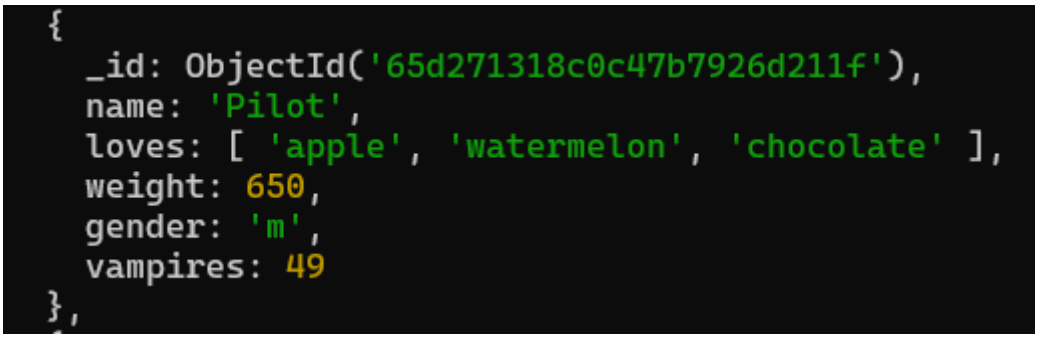
```
db.towns.updateOne({name:"Portland"}, {$unset:{"mayor.party":1}})
```

```
learn> db.towns.updateOne({name:"Portland"}, {$unset:{"mayor.party":1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find()
[
  {
    _id: ObjectId('65d27dba8c0c47b7926d2122'),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('65d27dce8c0c47b7926d2123'),
    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' }
  },
  {
    _id: ObjectId('65d27de08c0c47b7926d2124'),
    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.*
2. *Проверить содержимое коллекции* `unicorns`.

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



Практическое задание 3.3.7:

1. *Изменить информацию о самке единорога* `Aurora`: *теперь она любит еще и сахар, и лимоны.*
2. *Проверить содержимое коллекции* `unicorns`.

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

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.*

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

```
learn> db.towns.remove({"mayor.party":{$exists:false}})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 3 }
learn> db.towns.clear()
TypeError: db.towns.clear is not a function
learn> db.towns.remove()
MongoshInvalidInputError: [COMMON-10001] Missing required argument at position 0 (Collection.remove)
learn> db.towns.remove({})
{ acknowledged: true, deletedCount: 3 }
```

```
learn> db.getCollectionNames()
[ 'unicorns', 'towns' ]
```

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, использую второй способ автоматического связывания.*
3) *Проверьте содержание коллекции едиорогов.*
4) *Содержание коллекции единорогов* `unicorns`*:*

```
db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});

db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});

db.unicorns.insert({name: 'Roooooodles', 44), loves: ['apple'], weight: 575, gender: 'm', vampires: 99});

db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});

db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});

db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});

db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});

db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});

db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});

db.unicorns.insert ({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});

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

```
db.biomes.insert({_id:"desert", name:"long desert name", descriptions:"no water, a lot of sand"})

db.unicorns.update({_id:ObjectId('65d298415939b5e7af21a5ba')}, {"$set":{ habitat:{'$ref': 'biomes', '$id': 'desert' }}})
```

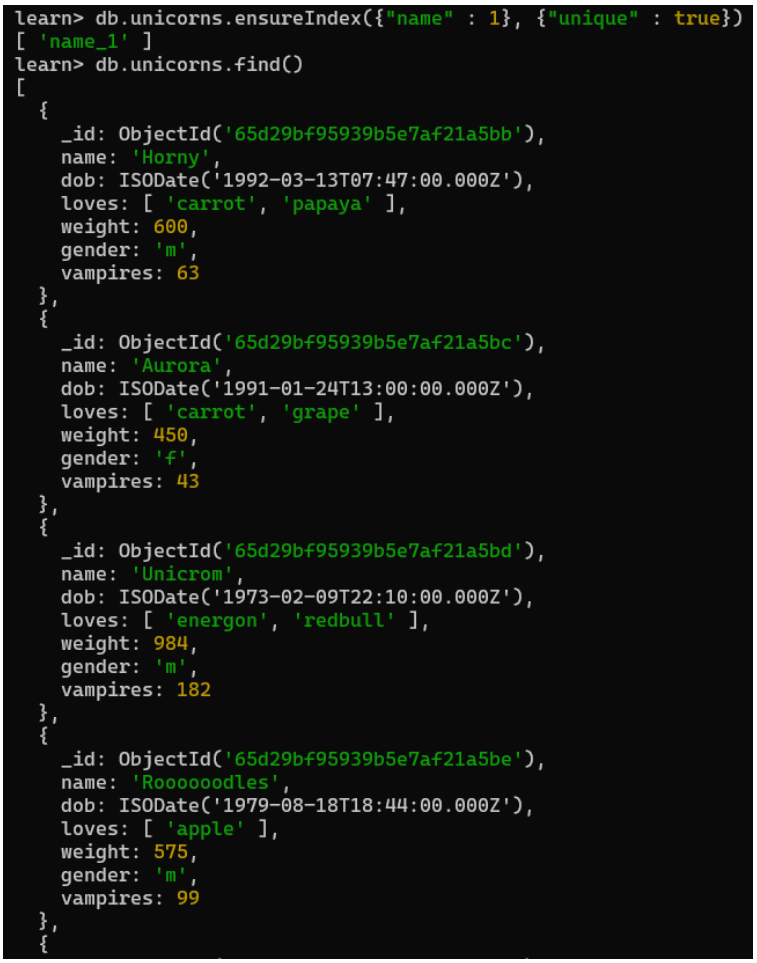
```
learn> db.unicorns.find({_id:ObjectId('65d298415939b5e7af21a5ba')})
[
  {
    _id: ObjectId('65d298415939b5e7af21a5ba'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99,
    habitat: DBRef('biomes', 'desert')
  }
]
```

Практическое задание 4.2.1:

1. *Проверьте, можно ли задать для коллекции* `unicorns` *индекс для ключа* `name` *с флагом* `unique`.
2. *Содержание коллекции единорогов* `unicorns`*:*

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

Можно



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.*

db.unicorns.getIndexes()

db.unicorns.dropIndex("name_1")

db.unicorns.dropIndex("_id_")

```
learn> db.unicorns.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
learn> db.dropIndex("name_1")
TypeError: db.dropIndex is not a function
learn> db.unicorn.dropIndex("name_1")
MongoServerError[NamespaceNotFound]: ns not found learn.unicorn
learn> db.unicorns.dropIndex("name_1")
{ nIndexesWas: 2, ok: 1 }
learn> db.unicorns.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn> db.unicorns.dropIndex("_id_")
MongoServerError[InvalidOptions]: cannot drop _id index
learn> s
```

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?*

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

```
learn> db.numbers.explain("executionStats").find().sort({$natural:-1}).limit(4)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: { stage: 'COLLSCAN', direction: 'backward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 1,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
```

Понадобилась одна миллисекунда.

Длительность показывает 0 миллисекунд, но явно это ошибка округления. Разницу между запросом с индексом и без определить очень сложно, потому что они выполняются очень быстро. Вывод об эффективности сделать не получается.

Вывод

Овладел 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.