

**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: Цатинян А.А.

Факультет: ПИН

Группа: К3239

Преподаватель: Говорова М.М.



Санкт-Петербург 2025

## **Оглавление**

Цель работы .....	3
Практическое задание.....	3
Выполнение.....	3
Вывод .....	29

## 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:

*Создайте базу данных learn.*

*Заполните коллекцию единорогов unicorns:*

```
> use learn
< switched to db 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('682f446d72f1c59f3d2b26e1'),
    '1': ObjectId('682f446d72f1c59f3d2b26e2'),
    '2': ObjectId('682f446d72f1c59f3d2b26e3'),
    '3': ObjectId('682f446d72f1c59f3d2b26e4'),
    '4': ObjectId('682f446d72f1c59f3d2b26e5'),
    '5': ObjectId('682f446d72f1c59f3d2b26e6'),
    '6': ObjectId('682f446d72f1c59f3d2b26e7'),
    '7': ObjectId('682f446d72f1c59f3d2b26e8'),
    '8': ObjectId('682f446d72f1c59f3d2b26e9'),
    '9': ObjectId('682f446d72f1c59f3d2b26ea'),
    '10': ObjectId('682f446d72f1c59f3d2b26eb')
  }
}
learn>
```

*Используя второй способ, вставьте в коллекцию единорогов документ:*

```
> let document = {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165};
db.unicorns.insertOne(document);
< {
  acknowledged: true,
  insertedId: ObjectId('682f452c72f1c59f3d2b26ec')
}
learn>
```

Проверьте содержимое коллекции с помощью метода `find`.

```
> db.unicorns.find().pretty()
< {
  _id: ObjectId('682f444d72f1c59f3d2b26d6'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26d7'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26d8'),
  name: 'Unicrom',
  loves: [
    'energon',
    'redbull'
  ],
  weight: 984,
  gender: 'm',
  vampires: 182
}
```

### Практическое задание 2.2.1:

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

Вывод списка самцов, отсортированных по имени

```
> db.unicorns.find({gender: 'm'}).sort({name: 1}).limit(3).pretty()
< {
  _id: ObjectId('682f452c72f1c59f3d2b26ec'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('682f446d72f1c59f3d2b26e1'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26d6'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
learn>
```

Вывод списка самцов, отсортированных по имени:

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3).pretty()
< {
  _id: ObjectId('682f446d72f1c59f3d2b26e2'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26d7'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26db'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 733,
  gender: 'f',
  vampires: 40
}
learn>
```

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

Поиск первой самки, любящей 'carrot', с помощью *findOne*

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
< {
  _id: ObjectId('682f444d72f1c59f3d2b26d7'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

learn>

Поиск первой самки, любящей 'carrot', с помощью *find().limit(1)*

```
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1).pretty()
< {
  _id: ObjectId('682f444d72f1c59f3d2b26d7'),
  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.

Вывод списка самцов (только имя, вес, вампиры, \_id), отсортированных по имени:

```
> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).sort({name: 1}).pretty()
< {
  _id: ObjectId('682f452c72f1c59f3d2b26ec'),
  name: 'Dunx',
  weight: 704,
  vampires: 165
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26d6'),
  name: 'Horny',
  weight: 600,
  vampires: 63
}
{
  _id: ObjectId('682f446d72f1c59f3d2b26e1'),
  name: 'Horny',
  weight: 600,
  vampires: 63
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26dc'),
  name: 'Kenny',
  weight: 690,
  vampires: 39
}
{
  _id: ObjectId('682f446d72f1c59f3d2b26e7'),
  name: 'Kenny',
  weight: 690,
  vampires: 39
}
```



### Практическое задание 2.2.3:

*Вывести список единорогов в обратном порядке добавления.*

Вывод списка единорогов в обратном порядке их добавления в коллекцию:

```
> db.unicorns.find().sort({$natural: -1}).pretty()
< {
  _id: ObjectId('682f452c72f1c59f3d2b26ec'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('682f446d72f1c59f3d2b26eb'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  _id: ObjectId('682f446d72f1c59f3d2b26ea'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 650,
  gender: 'm',
  vampires: 54
}
```

#### Практическое задание 2.2.4:

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

```
> db.unicorns.find({}, {name: 1, loves: {$slice: 1}, _id: 0}).pretty()
< {
  name: 'Horny',
  loves: [
    'carrot'
  ]
}
{
  name: 'Aurora',
  loves: [
    'carrot'
  ]
}
{
  name: 'Unicrom',
  loves: [
    'energon'
  ]
}
{
  name: 'Roooooodles',
  loves: [
    'apple'
  ]
}
{
  name: 'Solnara',
  loves: [
    'apple'
  ]
}
{
  name: 'Ayna',
  loves: [
    'strawberry'
  ]
}
```

### 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}).pretty()
< {
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],
  weight: 550,
  gender: 'f',
  vampires: 80
}
{
  name: 'Leia',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 601,
  gender: 'f',
  vampires: 33
}
{
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],
  weight: 550,
  gender: 'f',
  vampires: 80
}
```

### 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}).pretty()
< {
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
{
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
learn>
```

### Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ *vampires*.

```
> db.unicorns.find({vampires: {$exists: false}}).pretty()
< {
  _id: ObjectId('682f444d72f1c59f3d2b26e0'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  _id: ObjectId('682f446d72f1c59f3d2b26eb'),
  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: 1, loves: {$slice: 1}, _id: 0}).sort({name: 1}).pretty()
< {
  name: 'Dunx',
  loves: [
    'grape'
  ]
}
{
  name: 'Horny',
  loves: [
    'carrot'
  ]
}
{
  name: 'Horny',
  loves: [
    'carrot'
  ]
}
{
  name: 'Kenny',
  loves: [
    'grape'
  ]
}
{
  name: 'Kenny',
  loves: [
    'grape'
  ]
}
{
  name: 'Pilot',
  loves: [
```

### Практическое задание 3.1.1:

Создайте коллекцию *towns*, включающую документы

```
> db.towns.insertMany([
  {name: "Punxsutawney", population: 6200, last_census: ISODate("2008-01-31T00:00:00Z"), famous_for: [""], mayor: {name: "Jim Wehrle" }},
  {name: "New York", population: 22200000, last_census: ISODate("2009-07-31T00:00:00Z"), famous_for: ["statue of liberty", "food"], mayor: {name: "Michael Bloomberg", party: "I"}},
  {name: "Portland", population: 528000, last_census: ISODate("2009-07-20T00:00:00Z"), famous_for: ["beer", "food"], mayor: {name: "Sam Adams", party: "D"}}
]);
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('682f487072f1c59f3d2b26ed'),
    '1': ObjectId('682f487072f1c59f3d2b26ee'),
    '2': ObjectId('682f487072f1c59f3d2b26ef')
  }
}
```

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}).pretty()
< {
  name: 'New York',
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
learn>
```

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}).pretty()
< {
  name: 'Punxsutawney',
  mayor: {
    name: 'Jim Wehrle'
  }
}
learn> |
```

### Практическое задание 3.1.2:

Сформировать функцию для вывода списка самцов единорогов.

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

Вывести результат, используя `forEach`

```
> var printMaleUnicornsF = function() {
  var maleUnicornsCursor = db.unicorns.find({gender: 'm'}).sort({name: 1}).limit(2)
  maleUnicornsCursor.forEach(function(unicorn){
    printjson(unicorn);
  });
};
> printMaleUnicornsF();
< {
  _id: ObjectId('682f452c72f1c59f3d2b26ec'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
< {
  _id: ObjectId('682f446d72f1c59f3d2b26e1'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
learn>
```

### Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг

```
> db.unicorns.countDocuments({gender: 'f', weight: {$gte: 500, $lte: 600}})
< 4
learn> |
```

### Практическое задание 3.2.2:

Вывести список предпочтений.

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

### Практическое задание 3.2.3:

*Посчитать количество особей единорогов обоих полов.*

```
> db.unicorns.aggregate([
  {$group: {_id: "$gender", count: {$sum: 1}}}
]).pretty()
< {
  _id: 'f',
  count: 10
}
{
  _id: 'm',
  count: 13
}
learn>
```

### Практическое задание 3.3.1:

*Выполнить команду:*

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

*Проверить содержимое коллекции unicorns.*

```
> db.unicorns.insertOne({name: 'Barney', loves: ['grape'], weight: 340, gender: 'm'})
< {
  acknowledged: true,
  insertedId: ObjectId('682f4b2d72f1c59f3d2b26f0')
}
> db.unicorns.find({name: 'Barney'}).pretty()
< {
  _id: ObjectId('682f4b2d72f1c59f3d2b26f0'),
  name: 'Barney',
  loves: [
    'grape'
  ],
  weight: 340,
  gender: 'm'
}
learn> |
```



### Практическое задание 3.3.2:

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

Проверить содержимое коллекции *unicorns*.

```
> db.unicorns.updateOne({name: 'Ayna', gender: 'f'}, {$set: {weight: 800, vampires: 51}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.find({name: 'Ayna'}).pretty()
< {
  _id: ObjectId('682f444d72f1c59f3d2b26db'),
  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.

Проверить содержимое коллекции unicorns.

```
> db.unicorns.updateOne({name: 'Raleigh', gender: 'm'}, {$addToSet: {loves: 'redbull'}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.find({name: 'Raleigh'}).pretty()
< {
  _id: ObjectId('682f444d72f1c59f3d2b26dd'),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar',
    'redbull'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
```

### Практическое задание 3.3.4:

Всем самцам единорогов увеличить количество убитых вампиров на 5.

Проверить содержимое коллекции *unicorns*.

```
> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 14,
  modifiedCount: 14,
  upsertedCount: 0
}
> db.unicorns.find({gender: 'm'}).pretty()
< {
  _id: ObjectId('682f444d72f1c59f3d2b26d6'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 68
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26d8'),
  name: 'Unicrom',
  loves: [
    'energon',
    'redbull'
  ],
  weight: 984,
  gender: 'm',
  vampires: 187
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26d9'),
  name: 'Rooooooodles',
  loves: [
    'apple'
  ]
}
```

### Практическое задание 3.3.5:

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

Проверить содержимое коллекции towns.

```
> db.towns.updateOne({name: "Portland"}, {$unset: {"mayor.party": ""}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.towns.find({name: "Portland"}).pretty()
< {
  _id: ObjectId('682f487072f1c59f3d2b26ef'),
  name: 'Portland',
  population: 528000,
  last_census: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams'
  }
}
learn>
```

### Практическое задание 3.3.6:

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

Проверить содержимое коллекции *unicorns*.

```
> db.unicorns.updateOne({name: 'Pilot', gender: 'm'}, {$push: {loves: 'chocolate'}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.find({name: 'Pilot'}).pretty()
< {
  _id: ObjectId('682f444d72f1c59f3d2b26df'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
```

### Практическое задание 3.3.7:

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

Проверить содержимое коллекции unicorns.

```
> db.unicorns.updateOne(
  {name: 'Aurora'},
  {$addToSet: {loves: {$each: ['sugar', 'lemon']}}}
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.find({name: 'Aurora'}).pretty()
< {
  _id: ObjectId('682f444d72f1c59f3d2b26d7'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape',
    'sugar',
    'lemon'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

### Практическое задание 3.4.1:

Создайте коллекцию towns, включающую следующие документы

```
> db.towns.insertMany([
  {name: "Punxsutawney", population: 6200, last_census: ISODate("2008-01-31T00:00:00Z"), famous_for: ["phil the groundhog"], mayor: {name: "Jim Wehrle" }},
  {name: "New York", population: 22200000, last_census: ISODate("2009-07-31T00:00:00Z"), famous_for: ["status of liberty", "food"], mayor: {name: "Michael Bloomberg", party: "I"}},
  {name: "Portland", population: 528000, last_census: ISODate("2009-07-20T00:00:00Z"), famous_for: ["beer", "food"], mayor: {name: "Sam Adams", party: "D"}}
]);
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('682f588f72f1c59f3d2b26f1'),
    '1': ObjectId('682f588f72f1c59f3d2b26f2'),
    '2': ObjectId('682f588f72f1c59f3d2b26f3')
  }
}
learn>
```

Удалите документы с беспартийными мэрами.  
Проверьте содержание коллекции.

```
> db.towns.deleteMany({"mayor.party": "I"})
< {
  acknowledged: true,
  deletedCount: 1
}
> db.towns.find().pretty()
< {
  _id: ObjectId('682f58f672f1c59f3d2b26f4'),
  name: 'Punxsutawney',
  population: 6200,
  last_census: 2008-01-31T00:00:00.000Z,
  famous_for: [
    'phil the groundhog'
  ],
  mayor: {
    name: 'Jim Wehrle'
  }
}
{
  _id: ObjectId('682f58f672f1c59f3d2b26f6'),
  name: 'Portland',
  population: 528000,
  last_census: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams',
    party: 'D'
  }
}
learn>|
```

Очистите коллекцию.  
Просмотрите список доступных коллекций.

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

### Практическое задание 4.1.1:

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

```
> db.habitats.insertMany([
  { _id: "forest", name: "Enchanted Forest", description: "A mystical forest full of ancient trees and sparkling streams." },
  { _id: "mountain", name: "Crystal Mountains", description: "High peaks rumoured to hide unicorn treasures." },
  { _id: "meadow", name: "Flowering Meadow", description: "A vast meadow with a never-ending supply of sweet clover." }
]);
< {
  acknowledged: true,
  insertedIds: {
    '0': 'forest',
    '1': 'mountain',
    '2': 'meadow'
  }
}
learn>
```

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

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

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

db.unicorns.updateOne(
  { name: "Pilot" },
  { $set: { habitat: { $ref: "habitats", $id: "meadow", $db: "learn" } } }
);
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> |
```



Проверьте содержание коллекции едиорогов.

```
> db.unicorns.find({name: {$in: ["Aurora", "Horny", "Pilot"]}}).pretty()
< {
  _id: ObjectId('682f444d72f1c59f3d2b26d6'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 68,
  habitat: DBRef('habitats', 'mountain', 'learn')
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26d7'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape',
    'sugar',
    'lemon'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43,
  habitat: DBRef('habitats', 'forest', 'learn')
}
{
  _id: ObjectId('682f444d72f1c59f3d2b26df'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ]
}
```

#### **Практическое задание 4.2.1:**

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

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

### Практическое задание 4.3.1:

Получите информацию о всех индексах коллекции *unicorns*.

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

Удалите все индексы, кроме индекса для идентификатора.

```
> db.unicorns.dropIndex("name_1")
< { nIndexesWas: 2, ok: 1 }
learn>
```

Попытайтесь удалить индекс для идентификатора.

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

### Практическое задание 4.4.1:

Создайте объемную коллекцию *numbers*, задействовав курсор

```
> for(let i = 0; i < 1000000; i++){
  db.numbers.insertOne({value: i});
}
< {
  acknowledged: true,
  insertedId: ObjectId('682f5b3f72f1c59f3d2cada2')
}
```

Выберите последних четыре документа.

```
> db.numbers.find().sort({value: -1}).limit(4).pretty()
< {
  _id: ObjectId('682f5b3f72f1c59f3d2cada2'),
  value: 99999
}
{
  _id: ObjectId('682f5b3f72f1c59f3d2cada1'),
  value: 99998
}
{
  _id: ObjectId('682f5b3f72f1c59f3d2cada0'),
  value: 99997
}
{
  _id: ObjectId('682f5b3f72f1c59f3d2cad9f'),
  value: 99996
}
learn>|
```

Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса?

```
executionTimeMillis: 74,
```

Создайте индекс для ключа *value*.

```
> db.numbers.createIndex({value: 1})
< value_1
learn>
```

Получите информацию о всех индексах коллекции *numbers*.

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

Выполните запрос 2.

```
> db.numbers.find().sort({value: -1}).limit(4).pretty()
< {
  _id: ObjectId('682f5b3f72f1c59f3d2cada2'),
  value: 99999
}
{
  _id: ObjectId('682f5b3f72f1c59f3d2cada1'),
  value: 99998
}
{
  _id: ObjectId('682f5b3f72f1c59f3d2cada0'),
  value: 99997
}
{
  _id: ObjectId('682f5b3f72f1c59f3d2cad9f'),
  value: 99996
}
learn> |
```

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

```
executionTimeMillis: 8,
```

Запрос с использованием индекса по полю value выполняется значительно быстрее и, следовательно, более эффективен.

## **Вывод**

В процессе выполнения данных лабораторных работ были получены фундаментальные навыки работы с документ-ориентированной СУБД MongoDB, начиная с ее установки и первоначальной настройки, и заканчивая практическим применением CRUD-операций для манипулирования данными. Были освоены методы создания, выборки (с использованием различных операторов, проекций, сортировки и ограничения), обновления и удаления документов, включая работу с вложенными объектами и массивами. Кроме того, были изучены механизмы агрегирования данных, использование курсоров, создание ссылок между коллекциями и управление индексами, что позволило наглядно оценить их влияние на производительность запросов и заложило основу для эффективного использования MongoDB в разработке приложений.