

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6**  
**«ВВЕДЕНИЕ В СУБД MONGODB. УСТАНОВКА MONGODB. НАЧАЛО**  
**РАБОТЫ С БД»**  
**по дисциплине «Проектирование и реализация баз данных»**

**Обучающийся:** Бородин Максим Андреевич  
**Факультет** прикладной информатики  
**Группа** K3241  
**Направление подготовки** 09.03.03 Прикладная информатика  
**Образовательная программа** Мобильные и сетевые технологии 2023  
**Преподаватель** Говорова Марина Михайловна

Санкт-Петербург  
2025

## **Цель работы:**

Овладеть практическими навыками установки СУБД MongoDB.

## **Программное обеспечение**

СУБД MongoDB.

## **Практическое задание**

1. Установите MongoDB для обеих типов систем (32/64 бита).
2. Проверьте работоспособность системы запуском клиента mongo.
3. Выполните методы:
  - a. `db.help()`
  - b. `db.help`
  - c. `db.stats()`
4. Создайте БД learn.
5. Получите список доступных БД.
6. Создайте коллекцию unicorns, вставив в нее документ `{name: 'Aurora', gender: 'f', weight: 450}`.
7. Просмотрите список текущих коллекций.
8. Переименуйте коллекцию unicorns.
9. Просмотрите статистику коллекции.
10. Удалите коллекцию.
11. Удалите БД learn.

## **ЛАБОРАТОРНАЯ РАБОТА 6.2. Работа с БД в СУБД MongoDB**

**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.

### **2.1 ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ**

#### **Практическое задание 2.1.1**

1. Создайте базу данных learn.
2. Заполните коллекцию единорогов unicorns:
3. Используя второй способ, вставьте в коллекцию единорогов документ:
4. Проверьте содержимое коллекции с помощью метода find.

## Первый способ вставки

```
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('6825934d6ce661d53a6c4bd0') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('682593516ce661d53a6c4bd1') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('682593556ce661d53a6c4bd2') }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('682593596ce661d53a6c4bd3') }
}
learn> db.unicorns.insert({name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6825935d6ce661d53a6c4bd4') }
}
learn> db.unicorns.insert({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('682593606ce661d53a6c4bd5') }
}
learn> db.unicorns.insert({name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('682593646ce661d53a6c4bd6') }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('682593686ce661d53a6c4bd7') }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6825936c6ce661d53a6c4bd8') }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
```

## Второй способ вставки (через отдельную переменную)

```
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('682593fc6ce661d53a6c4bdb') }
}
learn>
```

## Проверка содержимого коллекции с помощью find:

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('6825934d6ce661d53a6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('682593516ce661d53a6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('682593556ce661d53a6c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('682593596ce661d53a6c4bd3'),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6825935d6ce661d53a6c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {

```

## **2.2 ВЫБОРКА ДАННЫХ ИЗ БД**

### **Практическое задание 2.2.1. Запросы для вывода списков самцов и самок**

1. Сформируйте запросы для вывода списков самцов и самок единорогов.  
Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.
2. Найдите всех самок, которые любят carrot. Ограничьте этот список первой особью с помощью функций `findOne` и `limit`.

Список самцов, отсортированный по имени:

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('682593fc6ce661d53a6c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6825934d6ce661d53a6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('682593646ce661d53a6c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('682593706ce661d53a6c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('682593686ce661d53a6c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('682593596ce661d53a6c4bd3'),
```

Список самок, ограниченный первыми тремя, отсортированный по имени:

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('682593516ce661d53a6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('682593606ce661d53a6c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('6825936c6ce661d53a6c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn>
```

Самки, любящие carrot, ограниченные первой особью:

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId('682593516ce661d53a6c4bd1'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```



### **Практическое задание 2.2.2. Модификация запроса для самцов без loves и gender**

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

Запрос для самцов без loves и gender:

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).sort({name: 1})
[
  {
    _id: ObjectId('682593fc6ce661d53a6c4bdb'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('6825934d6ce661d53a6c4bd0'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('682593646ce661d53a6c4bd6'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('682593706ce661d53a6c4bd9'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('682593686ce661d53a6c4bd7'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('682593596ce661d53a6c4bd3'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('682593556ce661d53a6c4bd2'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
]
learn>
```

### Практическое задание 2.2.3. Вывод списка единорогов в обратном порядке добавления

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('682593fc6ce661d53a6c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('682593746ce661d53a6c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('682593706ce661d53a6c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6825936c6ce661d53a6c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
  }
]
```

## Практическое задание 2.2.4. Вывод списка единорогов с первым любимым предпочтением, без \_id

```
learn> db.unicorns.find({}, {_id: 0, loves: {$slice: 1}})
[
  {
    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: 'Rooooooodles',
    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',
  }
]
```

## 2.3 ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

### Практическое задание 2.3.1 Вывод списка самок единорогов весом от полутонны до 700 кг, без \_id

```
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'
  }
]
learn>
```

### Практическое задание 2.3.2. Вывод списка самцов единорогов весом от полутонны и предпочитающих grape и lemon, без \_id

```
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
  }
]
learn>
```

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

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('682593746ce661d53a6c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn>
```

### Практическое задание 2.3.4 Вывод упорядоченного списка имён самцов единорогов с первым предпочтением

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

## 3.1 ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ

### Практическое задание 3.1.1 Создание коллекции towns и выполнение запросов к вложенным объектам

1. Создайте коллекцию towns, включающую следующие документы:
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.

Создание коллекции towns и вставка документов:

```
towns> db.towns.insertMany([ { name: "Punxsutawney", populatiuon: 6200, last_sensus: ISODate("2008-01-31"), famous_f
31"), famous_for: ["status of liberty", "food"], mayor: { name: "Michael Bloomberg", party: "I" } }, { name: "Portla
s", party: "D" } } ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68259a6700f69f3a0a6c4bd0'),
    '1': ObjectId('68259a6700f69f3a0a6c4bd1'),
    '2': ObjectId('68259a6700f69f3a0a6c4bd2')
  }
}
towns> █
```

Запрос для городов с независимыми мэрами (party="I"):

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

Запрос для городов с беспартийными мэрами (party отсутствует):

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

### Практическое задание 3.1.2 Создание функции и работа с курсором для единорогов

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`

Создание функции для вывода списка самцов единорогов:

```
test> use unicorns
switched to db unicorns
unicorns> function getMaleUnicorns() {return db.unicorns.find({ gender: 'm' });}
[Function: getMaleUnicorns]
unicorns> █
```

Создание курсора для первых двух самцов с сортировкой по имени:

```
unicorns> var cursor = getMaleUnicorns().sort({name: 1}).limit(2)
unicorns> █
```

Вывод результата с помощью `forEach`:



```
unicorns> cursor.forEach(function (doc) { printjson(doc); });
{
  _id: ObjectId('68259c6d217556623c6c4bda'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('68259c6d217556623c6c4bd0'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}

unicorns>
```

## 3.2 АГРЕГИРОВАННЫЕ ЗАПРОСЫ

**Практическое задание 3.2.1: вывести количество самок единорогов весом от полутонны до 600 кг.**

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

**Практическое задание 3.2.2: вывести список предпочтений.**

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

**Практическое задание 3.2.3: посчитать количество особей единорогов обоих полов.**

```
unicorns> db.unicorns.aggregate([{$group: {_id: "$gender", count: {$sum: 1}}}]  
[ { _id: 'f', count: 5 }, { _id: 'm', count: 6 } ]  
unicorns>
```

**Практическое задание 3.3.1:**

Выполнить команду:

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

Метод `save()` был удалён из новых MongoDB-драйверов и `mongosh`.

**Практическое задание 3.3.2: для самки единорога Айна внести изменения в БД: теперь ее вес 800, она убила 51 вампира.**

1. Для самки единорога Айна внести изменения в БД: теперь ее вес 800, она убила 51 вампира.
2. Проверить содержимое коллекции unicorns.

```
unicorns> db.unicorns.updateOne({name: "Ayna"}, {$set: {weight: 800, vampires: 51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
unicorns> db.unicorns.find({name: "Ayna"})
[
  {
    _id: ObjectId('68259c6d217556623c6c4bd4'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
unicorns>
```

**Практическое задание 3.3.3: для самца единорога Raleigh внести изменения в БД: теперь он любит рэдбул.**

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

```
unicorns> db.unicorns.find({name: "Raleigh"})
[
  {
    _id: ObjectId('68259c6d217556623c6c4bd6'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
unicorns> db.unicorns.updateOne({name: "Raleigh"}, {$set: {loves: ["redbull"]}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
unicorns> db.unicorns.find({name: "Raleigh"})
[
  {
    _id: ObjectId('68259c6d217556623c6c4bd6'),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
unicorns>
```

### Практическое задание 3.3.4: Всем самцам единорогов увеличить количество убитых вампиров на 5.

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.
2. Проверить содержимое коллекции unicorns.

```
unicorns> db.unicorns.updateMany({gender: "m"}, {$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 6,
  modifiedCount: 6,
  upsertedCount: 0
}
unicorns> db.unicorns.find({gender: "m"})
[
  {
    _id: ObjectId('68259c6d217556623c6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('68259c6d217556623c6c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId('68259c6d217556623c6c4bd5'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 44
  },
  {
    _id: ObjectId('68259c6d217556623c6c4bd6'),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 7
  },
  {
    _id: ObjectId('68259c6d217556623c6c4bd8'),
```

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

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

```
towns> db.towns.updateOne({name: "Portland"}, {$unset: {"mayor.party": 1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
towns> db.towns.find(name: "Portland")
Uncaught:
SyntaxError: Unexpected token, expected "," (1:18)

> 1 | db.towns.find(name: "Portland")
    |                               ^
    2 |

towns> db.towns.find({name: "Portland"})
[
  {
    _id: ObjectId('68259a6700f69f3a0a6c4bd2'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
towns> S
```

### Практическое задание 3.3.6: изменить информацию о самце единорога

**Pilot:** теперь он любит и шоколад.

1. Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.
2. Проверить содержимое коллекции unicorns.

```
towns> use unicorns
switched to db unicorns
unicorns> db.unicorns.find({name: "Pilot"})
[
  {
    _id: ObjectId('68259c6d217556623c6c4bd8'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
unicorns> db.unicorns.updateOne({name: "Pilot"}, {$push: {loves: "chocolate"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
unicorns> db.unicorns.find({name: "Pilot"})
[
  {
    _id: ObjectId('68259c6d217556623c6c4bd8'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
unicorns> █
```

### Практическое задание 3.3.7: изменить информацию о самке единорога

**Aurora: теперь она любит еще и сахар, и лимоны.**

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

```
unicorns> db.unicorns.find({name: "Aurora"})
[
  {
    _id: ObjectId('68259c6d217556623c6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
unicorns> db.unicorns.updateOne({name: "Aurora"}, {$addToSet: {loves: {$each: ["sugar", "lemon"]}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
unicorns> db.unicorns.find({name: "Aurora"})
[
  {
    _id: ObjectId('68259c6d217556623c6c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
unicorns> █
```



## 3.4 УДАЛЕНИЕ ДАННЫХ ИЗ КОЛЛЕКЦИИ

### Практическое задание 3.4.1:

1. Создайте коллекцию towns, включающую следующие документы: ...
2. Удалите документы с беспартийными мэрами.
3. Проверьте содержание коллекции.
4. Очистите коллекцию.
5. Просмотрите список доступных коллекций.

```
towns> db.towns.insertMany([{name: "Punxsutawney ", popujatiuon: 6200, last_sensus: ISODate("2009-07-31"), famous_for: ["status of liberty", "food"], mayor: { name: "Michael Bloomberg", party: "D"}}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6825a671217556623c6c4bdb'),
    '1': ObjectId('6825a671217556623c6c4bdc'),
    '2': ObjectId('6825a671217556623c6c4bdd')
  }
}
towns>
```

### Удаление беспартийных меров

```
towns> db.towns.deleteMany({"mayor.party": {"$exists": 0}})
{ acknowledged: true, deletedCount: 1 }
towns> db.towns.find()
[
  {
    _id: ObjectId('6825a6d9217556623c6c4bdf'),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('6825a6d9217556623c6c4be0'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
towns>
```

## Отчистка коллекции

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

## 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.
2. Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания.
3. Проверьте содержание коллекции единорогов.

### Создание коллекции зон обитания

```
towns> use zones
switched to db zones
zones> db.zones.insertMany([{_id: "forest", name: "big forest", decription: "forest decs"}, {_id: "field", name: "big field", description: "field desc"}])
{ acknowledged: true, insertedIds: { '0': 'forest', '1': 'field' } }
zones>
```

### Добавление ссылки на зону

```
unicorns> db.unicorns.updateOne({name: "Dunx"}, {$set: {zone: {$ref: "zones", $id: "field"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
unicorns> db.unicorns.updateOne({name: "Nimue"}, {$set: {zone: {$ref: "zones", $id: "field"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
unicorns> db.unicorns.updateOne({name: "Pilot"}, {$set: {zone: {$ref: "zones", $id: "field"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
unicorns>
```

Результат:

```
_id: ObjectId('68259c6d217556623c6c4bd8'),
name: 'Pilot',
loves: [ 'apple', 'watermelon', 'chocolate' ],
weight: 650,
gender: 'm',
vampires: 59,
zone: DBRef('zones', 'field')
},
{
  _id: ObjectId('68259c6d217556623c6c4bd9'),
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f',
  zone: DBRef('zones', 'field')
},
{
  _id: ObjectId('68259c6d217556623c6c4bda'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 170,
  zone: DBRef('zones', 'field')
}
]
unicorns>
```

## 4.2 НАСТРОЙКА ИНДЕКСОВ

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

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

## 4.3 УПРАВЛЕНИЕ ИНДЕКСАМИ

**Практическое задание 4.3.1:**

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

```
unicorns> db.unicorns.ensureIndex({name: 1}, {unique: 1})
[ 'name_1' ]
unicorns> db.unicorns.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
unicorns> db.unicorns.dropIndexes()
{
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
unicorns> db.unicorns.dropIndex("_id_")
TypeError: db.unicorns.dropIndex is not a function
unicorns> █
```

## 4.4 ПЛАН ЗАПРОСА

### Практическое задание 4.4.1:

1. Создайте объемную коллекцию numbers, задействовав курсор:
2. `for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}`
3. Выберите последних четыре документа.
4. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра `executionTimeMillis`)
5. Создайте индекс для ключа `value`.
6. Получите информацию о всех индексах коллекции `numbers`.
7. Выполните запрос 2.
8. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?
9. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

```
unicorns> use numbers
switched to db numbers
numbers> for (i = 0; i < 100000; i++){db.numbers.insert({value: i})}

{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6825ad12217556623c6dd280') }
}
numbers>

numbers>
```

```
numbers> db.numbers.find().sort({_id: -1}).limit(4)
[
  { _id: ObjectId('6825ad12217556623c6dd280'), value: 99999 },
  { _id: ObjectId('6825ad12217556623c6dd27f'), value: 99998 },
  { _id: ObjectId('6825ad12217556623c6dd27e'), value: 99997 },
  { _id: ObjectId('6825ad12217556623c6dd27d'), value: 99996 }
]
numbers>
```



```

numbers> db.numbers.explain("executionStats").find({value: 99999})
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'numbers.numbers',
    parsedQuery: { value: { '$eq': 99999 } },
    indexFilterSet: false,
    queryHash: 'FBBD8DD0',
    planCacheShapeHash: 'FBBD8DD0',
    planCacheKey: 'CCC7FD70',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { value: { '$eq': 99999 } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 27,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { value: { '$eq': 99999 } },
      nReturned: 1,
      executionTimeMillisEstimate: 24,
      works: 100001,
      advanced: 1,
      needTime: 99999,
      needYield: 0,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 100000
    }
  }
}

```

```

numbers> db.numbers.createIndex({value: 1})
value_1
numbers> db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_ '},
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
numbers> █

```

```

numbers> db.numbers.find().sort({_id: -1}).limit(4)
[
  { _id: ObjectId('6825ad12217556623c6dd280'), value: 99999 },
  { _id: ObjectId('6825ad12217556623c6dd27f'), value: 99998 },
  { _id: ObjectId('6825ad12217556623c6dd27e'), value: 99997 },
  { _id: ObjectId('6825ad12217556623c6dd27d'), value: 99996 }
]
numbers> db.numbers.explain("executionStats").find({value: 99999})
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'numbers.numbers',
    parsedQuery: { value: { '$eq': 99999 } },
    indexFilterSet: false,
    queryHash: 'FBBD8DD0',
    planCacheShapeHash: 'FBBD8DD0',
    planCacheKey: '463AB5A3',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { value: 1 },
        indexName: 'value_1',
        isMultiKey: false,
        multiKeyPaths: { value: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { value: [ '[99999, 99999]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 6,
    totalKeysExamined: 1,
    totalDocsExamined: 1,
    executionStages: {
      isCached: false,

```

**Запрос выполнялся быстрее**



## **Вывод**

В процессе выполнения лабораторной работы были изучены основные операции в MongoDB. Получены навыки связывания коллекций, использования операторов \$set, \$unset, \$push, \$addToSet, \$inc, работы с индексами, а также анализа выполнения запросов с помощью explain.