

**Министерство науки и высшего образования Российской  
Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ  
АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ОБРАЗОВАНИЯ  
«Национальный исследовательский университет ИТМО»  
(Университет ИТМО)**

**Факультет прикладной информатики  
Образовательная программа Мобильные и сетевые технологии  
Направление подготовки 09.03.03 Мобильные и сетевые технологии**

**О Т Ч Е Т**

**Лабораторная работа № 6.**

**Тема работы: «Работа с БД в СУБД MongoDB»**

**Обучающийся: Кошкарев Кирилл Павлович, К3239**

**Преподаватель: Говорова М. М.**

**Санкт-Петербург  
2025**

## Цель

Изучить основные операции в СУБД MongoDB, включая создание базы данных, коллекций, вставку и изменение документов, а также базовую работу с индексами и запросами.

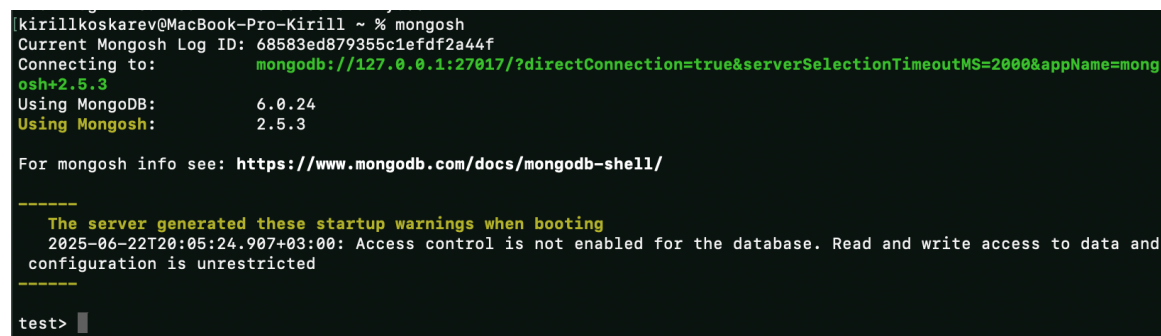
## Ход работы

### 1. Установка и запуск MongoDB

MongoDB был установлен через Homebrew командой:

```
brew install mongodb-community  
brew services start mongodb/brew/mongodb-community
```

Проверка подключения через оболочку: mongosh



```
kirillkoskarev@MacBook-Pro-Kirill ~ % mongosh  
Current Mongosh Log ID: 68583ed879355c1efdf2a44f  
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.3  
Using MongoDB:      6.0.24  
Using Mongosh:       2.5.3  
  
For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/  
  
-----  
The server generated these startup warnings when booting  
2025-06-22T20:05:24.907+03:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted  
-----  
test>
```

Рис. 1: Успешный запуск mongosh

### 2. Создание базы данных и коллекции

```
use learn  
db.unicorns.insert({...}) ...
```

```

test> use learn
switched to db learn
learn> 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: 'Rooooooodles', 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: 23});
... 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'});
...
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('682b17ceeb352b92c0606af7') }
}
learn>

```

Рис. 2: Заполнение коллекции

## Вставка документа (второй способ)

```

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})
... db.unicorns.insert(document)
...
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('682b19a2eb352b92c0606af8') }
}
learn>

```

Рис. 3: Заполнение коллекции

## 2.2 Выборка данных

```
db.unicorns.find()
```

### 2.2.1 Выборка документов по полу и предпочтениям

```

// Самцы (отсортировано по имени) db.unicorns.find({gender:
"m"}).sort({name: 1})

```

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('682b17ceeb352b92c0606aed'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606aee'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606aef'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af0'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af1'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

Рис. 4: Пример выборки

```
// Самки (только первые 3 по имени)
db.unicorns.find({gender: "f"}).sort({name: 1}).limit(3)

// Самка, любящая carrot (первый найденный документ)
db.unicorns.findOne({gender: "f", loves: "carrot"})

// Альтернатива через find + limit
db.unicorns.find({gender: "f", loves: "carrot"}).limit(1)
```

### 2.2.2 Исключение полей из выборки

```
// Самцы, без loves и gender db.unicorns.find(
  { gender: "m" },
```

```
learn> db.unicorns.find({gender: "m"}).sort({name: 1})
[...
[
  {
    _id: ObjectId('682b19a2eb352b92c0606af8'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606aed'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af3'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af6'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
]
```

Рис. 5: Самцы (отсортировано по имени)

```
{ loves: 0, gender: 0 }
)
```

### 2.2.3 Обратный порядок добавления

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

### 2.2.4 Вывод имени и первого предпочтения

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

```
learn> db.unicorns.find({gender: "f"}).sort({name: 1}).limit(3)
...
[
  {
    _id: ObjectId('682b17ceeb352b92c0606aee'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af2'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af5'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

Рис. 6: Самки (только первые 3 по имени)

```
learn> db.unicorns.findOne({gender: "f", loves: "carrot"})
{
  _id: ObjectId('682b17ceeb352b92c0606aee'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Рис. 7: Самка, любящая carrot (первый найденный документ)

### 2.3.1 Самки весом от 500 до 700 кг

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

### 2.3.2 Самцы, любящие grape и lemon, весом от 500 кг

```
db.unicorns.find(
```

```
learn> db.unicorns.find({gender: "f", loves: "carrot"}).limit(1)
...
[
  {
    _id: ObjectId('682b17ceeb352b92c0606aee'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Рис. 8: Альтернатива через find + limit

```
{ gender: "m", weight: { $gte: 500 }, loves: {
  $all: ["grape", "lemon"] }
},
{
  _id: 0
}
)
```

### 2.3.3 Единороги без поля vampires

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

### 2.3.4 Список имён самцов и первое предпочтение

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

```

learn> db.unicorns.find(
...   { gender: "m" },
...   { loves: 0, gender: 0 }
... )
[...
[
  {
    _id: ObjectId('682b17ceeb352b92c0606aed'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606aef'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af0'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af3'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
]

```

Рис. 9: Самцы без инфы о поле и предпочтениях

### 3.1.1 Работа с вложенными объектами

// Добавление городов db.towns.insertMany([...])

```

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

```

```

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

```



```
learn> db.unicorns.find().sort({$natural: -1})
...
[
  {
    _id: ObjectId('682b19a2eb352b92c0606af8'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af7'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af6'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('682b17ceeb352b92c0606af5'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],

```

Рис. 10: Все в обратном порядке добавления(новые сверху)

### 3.1.2 Использование курсора и forEach

```
// Создание курсора: первые два самца по алфавиту
var cursor = db.unicorns.find(
  { gender: "m" },
  { _id: 0, name: 1 }
).sort({ name: 1 }).limit(2)

// Обход курсора с forEach
cursor.forEach(function(unicorn) {
  print(unicorn.name)
})
```

### 3.2.1 Количество самок с весом 500–600 кг

```
db.unicorns.countDocuments({
  gender: "f", weight: { $gte: 500, $lte:
    600 }
```

}}

```
learn> db.unicorns.find(
...   {},
...   {
...     _id: 0,
...     name: 1,
...     loves: { $slice: 1 }
...   }
... )
[...
[
  { name: 'Horny', loves: [ 'carrot' ] },
  { name: 'Aurora', loves: [ 'carrot' ] },
  { name: 'Unicrom', loves: [ 'energon' ] },
  { name: 'Rooooooodles', loves: [ 'apple' ] },
  { name: 'Solnara', loves: [ 'apple' ] },
  { name: 'Ayna', loves: [ 'strawberry' ] },
  { name: 'Kenny', loves: [ 'grape' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Leia', loves: [ 'apple' ] },
  { name: 'Pilot', loves: [ 'apple' ] },
  { name: 'Nimue', loves: [ 'grape' ] },
  { name: 'Dunx', loves: [ 'grape' ] }
]
learn> |
```

Рис. 11: Все в без id с первым предпочтением

### 3.2.2 Список уникальных предпочтений

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

### 3.2.3 Количество особей по полу

```
db.unicorns.aggregate([
  {
    $group: { _id: "$gender",
              count: { $sum: 1 }
            }
  }
])
```

### 3.3.1 Вставка Barny (аналог save)

```
db.unicorns.insertOne({
```

name: "Barney", loves:  
["grape"],

```
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'
  }
]
```

Рис. 12: Самки от 500 о 700 кг

weight: 340,  
gender: "m" })  
db.unicorns.find({ name: "Barney" })

### 3.3.2 Обновление данных для Айна

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

### 3.3.3 Обновление предпочтений Raleigh

```
db.unicorns.updateOne( {
  name: "Raleigh" },
```

```
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
  }
]
```

Рис. 13: Самцы, любящие grape и lemon, весом от 500 кг

```
{
  $push: { loves: "redbull" }
}) db.unicorns.find({ name: "Raleigh" })
```

### 3.3.4 Увеличение количества вампиров у самцов

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

db.unicorns.find(
  { gender: "m" },
  { _id: 0, name: 1, vampires: 1 }
)
```

### 3.3.5 Удаление информации о партии мэра Portland

```
db.towns.updateOne(
```

```
learn> db.unicorns.find(
...   {
...     vampires: { $exists: false }
...   }
... )
[...
[
  {
    _id: ObjectId('682b17ceeb352b92c0606af7'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
]
```

Рис. 14: Единороги без поля vampires

```
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: 'Roooooodles', loves: [ 'apple' ] },
  { name: 'Unicrom', loves: [ 'energon' ] }
]
]
```

Рис. 15: Список имён самцов и первое предпочтение

```
{ name: "Portland" },
{ $unset: { "mayor.party": 1 } }
)

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

```

learn> db.towns.insertMany([
...   {
...     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"
...     }
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('682b63b5eb352b92c0606af9'),
    '1': ObjectId('682b63b5eb352b92c0606afa'),
    '2': ObjectId('682b63b5eb352b92c0606afb')
  }
}

```

Рис. 16: Добавление городов

### 3.3.6 Добавление шоколада к предпочтениям Pilot

```

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

```

```

db.unicorns.find(
  { name: "Pilot" },
  { _id: 0, name: 1, loves: 1 }
)

```

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

Рис. 17: Независимые мэры

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

Рис. 18: Беспартийные мэры

### 3.3.7 Добавление sugar и lemons к предпочтениям Aurora

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

db.unicorns.find(
  { name: "Aurora" },
  { _id: 0, name: 1, loves: 1 }
)
```

### 3.4.1 Удаление и очистка коллекции towns 3.4.1

#### Создание и очистка коллекции towns

```
// Вставка городов db.towns.insertMany([
  { name: "Punxsutawney",
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31"), famous_for:
    ["phil the groundhog"], mayor: { name: "Jim
    Wehrle" }
```

```

learn> function isMale() {
...   return this.gender === "m";
... }
[...
[Function: isMale]
learn> var cursor = db.unicorns.find(
...   { gender: "m" },
...   { _id: 0, name: 1 }
... ).sort({ name: 1 }).limit(2)
[...

learn> cursor.forEach(function(unicorn) {
...   print(unicorn.name)
... })
[...
Dunx
Horny

```

Рис. 19: Создание курсора и его обход с forEach

```

},
{ 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" }
}
])

```

```

// Удалить беспартийных мэров
db.towns.deleteMany({ "mayor.party": { $exists: false } })

```

```

// Проверить содержимое
db.towns.find().pretty() //

```

Очистить коллекцию



```

learn> db.unicorns.countDocuments({
...   gender: "f",
...   weight: { $gte: 500, $lte: 600 }
... })
[...
2
learn> db.unicorns.distinct("loves")
[...
[
  'apple',      'carrot',
  'chocolate', 'energon',
  'grape',      'lemon',
  'papaya',     'redbull',
  'strawberry', 'sugar',
  'watermelon'
]
learn> db.unicorns.aggregate([
...   {
...     $group: {
...       _id: "$gender",
...       count: { $sum: 1 }
...     }
...   }
... ])
[...
[ { _id: 'm', count: 7 }, { _id: 'f', count: 5 } ]
learn> db.unicorns.save({
...   name: "Barny",
...   loves: ["grape"],
...   weight: 340,
...   gender: "m"
... })
[...
TypeError: db.unicorns.save is not a function

```

Рис. 20: Задания 3.2.1-3.2.3 + 3.3.1

```
db.towns.deleteMany({})
```

```
// Показать список коллекций show
collections
```

#### 4.1.1 Зоны обитания и ссылки на них (DBRef)

```

// Создание зон обитания db.habitats.insertMany([
{
  _id: "forest", name:
    "Enchanted Forest",
    description: "Волшебный лес с сияющими растениями"
},
{

```

```
learn> db.unicorns.updateOne(
...   { name: "Ayna" },
...   {
...     $set: {
...       weight: 800,
...       vampires: 51
...     }
...   }
... )
[...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
]
learn> db.unicorns.find({ name: "Ayna" })
[...
[
  {
    _id: ObjectId('682b17ceeb352b92c0606af2'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
]
```

Рис. 21: Задание 3.3.2

```
_id: "mountains", name: "Crystal Mountains", description: "Горная
местность, где рождаются радуги"
},
{
  _id: "valley", name:
  "Golden Valley",
  description: "Уютная долина, полная мирных созданий" }
])

// Привязка единорогов db.unicorns.updateOne(
  { name: "Aurora" },
  { $set: { habitat: { $ref: "habitats", $id: "forest" } } }
)
```

```

learn> db.unicorns.updateOne(
...   { name: "Raleigh" },
...   {
...     $push: { loves: "redbull" }
...   }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Raleigh" })
...
[
  {
    _id: ObjectId('682b17ceeb352b92c0606af4'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]

```

Рис. 22: Задание 3.3.3

```

db.unicorns.updateOne(
  { name: "Pilot" },
  { $set: { habitat: { $ref: "habitats", $id: "mountains" } } }
)

```

```

db.unicorns.updateOne(
  { name: "Leia" },
  { $set: { habitat: { $ref: "habitats", $id: "valley" } } }
)

```

```

// Проверка db.unicorns.find(
  { habitat: { $exists: true } },
  { _id: 0, name: 1, habitat: 1 }
)

```

```

learn> db.unicorns.updateMany(
...   { gender: "m" },
...   { $inc: { vampires: 5 } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
learn> db.unicorns.find(
...   { gender: "m" },
...   { _id: 0, name: 1, vampires: 1 }
... )
...
[
  { name: 'Horny', vampires: 68 },
  { name: 'Unicrom', vampires: 187 },
  { name: 'Rooooooodles', vampires: 104 },
  { name: 'Kenny', vampires: 44 },
  { name: 'Raleigh', vampires: 7 },
  { name: 'Pilot', vampires: 59 },
  { name: 'Dunx', vampires: 170 },
  { name: 'Barney', vampires: 5 }
]

```

Рис. 23: Задание 3.3.4

#### 4.2.1 Создание уникального индекса по полю name

```

// Попытка создать уникальный индекс db.unicorns.createIndex(
  { name: 1 },
  { unique: true }
)

```

```

// Проверка списка индексов db.unicorns.getIndexes()

```

#### 4.3.1 Удаление индексов в коллекции unicorns

```

// Получение списка индексов db.unicorns.getIndexes()

```

```

// Удаление индекса по name (если он был создан)
db.unicorns.dropIndex("name_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(
...   { name: "Portland" },
...   { _id: 0, name: 1, mayor: 1 }
... )
...
[ { name: 'Portland', mayor: { name: 'Sam Adams' } } ]

```

Рис. 24: Задание 3.3.5

```

learn> db.unicorns.updateOne(
...   { name: "Pilot" },
...   { $push: { loves: "chocolate" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find(
...   { name: "Pilot" },
...   { _id: 0, name: 1, loves: 1 }
... )
...
[ { name: 'Pilot', loves: [ 'apple', 'watermelon', 'chocolate' ] } ]

```

Рис. 25: Задание 3.3.6

// Попытка удалить индекс `_id` (ожидается ошибка) `db.unicorns.dropIndex("_id_")`

#### 4.4.1 Альтернативный анализ производительности

// Без индекса

`db.numbers.explain("executionStats").find({ value: 99999 })`

// Создание индекса `db.numbers.createIndex({ value: 1 })`

// С индексом

`db.numbers.explain("executionStats").find({ value: 99999 })`

Запрос с индексом выполняется быстрее, а значит он более эффективен, чем первый запрос.

```
learn> db.unicorns.updateOne(
...   { name: "Aurora" },
...   { $addToSet: { loves: { $each: ["sugar", "lemons"] } } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find(
...   { name: "Aurora" },
...   { _id: 0, name: 1, loves: 1 }
... )
...
[ { name: 'Aurora', loves: [ 'carrot', 'grape', 'sugar', 'lemons' ] } ]
```

Рис. 26: Задание 3.3.7

## 1 Вывод

В ходе выполнения лабораторной работы осуществлялось знакомство с документо-ориентированной СУБД MongoDB через практическое применение её инструментов. Была проведена серия операций, включающая добавление новых документов, их редактирование и удаление, а также манипуляции с вложенными структурами и массивами. Отдельный акцент сделан на создании индексов и оценке их влияния на производительность запросов с помощью анализа `explain`. Также были протестированы способы установления связей между коллекциями и применены операторы, позволяющие гибко изменять данные (`set`, `unset`, `push`, `addToSet`, `inc`). Полученный опыт способствовал формированию глубокого понимания особенностей MongoDB и укреплению практических навыков работы с её механизмами.

```

learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     populatioun: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: ["phil the groundhog"],
...     mayor: {
...       name: "Jim Wehrle"
...     }
...   },
...   {
...     name: "New York",
...     populatioun: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["status of liberty", "food"],
...     mayor: {
...       name: "Michael Bloomberg",
...       party: "I"
...     }
...   },
...   {
...     name: "Portland",
...     populatioun: 528000,
...     last_sensus: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: {
...       name: "Sam Adams",
...       party: "D"
...     }
...   }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('682b6b90eb352b92c0606afd'),
    '1': ObjectId('682b6b90eb352b92c0606afe'),
    '2': ObjectId('682b6b90eb352b92c0606aff')
  }
}
learn> db.towns.deleteMany({
...   "mayor.party": { $exists: false }
... })
...
{ acknowledged: true, deletedCount: 1 }

```

Рис. 27: Задание 3.4.1 часть 1

```

{ acknowledged: true, deletedCount: 2 }
learn> db.towns.find().pretty()
...
[
  {
    _id: ObjectId('682b6b90eb352b92c0606afe'),
    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('682b6b90eb352b92c0606aff'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn> db.towns.deleteMany({})
...
{ acknowledged: true, deletedCount: 2 }
learn> show collections
...
towns
unicorns

```

Рис. 28: Задание 3.4.1 часть 2

```

unicorns
learn> db.habitats.insertMany([
...   {
...     _id: "forest",
...     name: "Enchanted Forest",
...     description: "Волшебный лес с сияющими растениями"
...   },
...   {
...     _id: "mountains",
...     name: "Crystal Mountains",
...     description: "Горная местность, где рождаются радуги"
...   },
...   {
...     _id: "valley",
...     name: "Golden Valley",
...     description: "Уютная долина, полная мирных созданий"
...   }
... ])
...
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'mountains', '2': 'valley' }
}
learn> db.unicorns.updateOne(
...   { name: "Aurora" },
...   { $set: { habitat: { $ref: "habitats", $id: "forest" } } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne(
...   { name: "Pilot" },
...   { $set: { habitat: { $ref: "habitats", $id: "mountains" } } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

```

Рис. 29: Задание 4.1.1 часть 1



```

learn> db.unicorns.updateOne(
...   { name: "Pilot" },
...   { $set: { habitat: { $ref: "habitats", $id: "mountains" } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne(
...   { name: "Leia" },
...   { $set: { habitat: { $ref: "habitats", $id: "valley" } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find(
...   { habitat: { $exists: true } },
...   { _id: 0, name: 1, habitat: 1 }
... )
...
[
  { name: 'Aurora', habitat: DBRef('habitats', 'forest') },
  { name: 'Leia', habitat: DBRef('habitats', 'valley') },
  { name: 'Pilot', habitat: DBRef('habitats', 'mountains') }
]

```

Рис. 30: Задание 4.1.1 часть 2

```

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

```

Рис. 31: Задание 4.2.1

```

learn> db.unicorns.getIndexes()
...
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
learn> db.unicorns.dropIndex("name_1")
...
{ nIndexesWas: 2, ok: 1 }
learn> db.unicorns.dropIndex("_id")
...
MongoServerError[InvalidOptions]: cannot drop _id index

```

Рис. 32: Задание 4.3.1

```
executionStats: {  
  executionSuccess: true,  
  nReturned: 1,  
  executionTimeMillis: 43,  
  totalKeysExamined: 0,  
  totalDocsExamined: 100000
```

Рис. 33: Задание 4.4.1 до создания индекса

```
executionStats: {  
  executionSuccess: true,  
  nReturned: 1,  
  executionTimeMillis: 3,  
  totalKeysExamined: 1,  
  totalDocsExamined: 1
```

Рис. 34: Задание 4.4.1 после создания индекса