Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«Работа с БД в СУБД MongoDB»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Недиков Михаил Олегович

Факультет прикладной информатики

Группа K3239

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2025

Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2025/2026

1. 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.

2. Практическое задание

1. 2 CRUD-ОПЕРАЦИИ В СУБД MONGODB. ВСТАВКА ДАННЫХ. ВЫБОРКА ДАННЫХ
2. ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB.ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ
3. ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

3. Выполнение

Практическое задание 2.1.1

Формулировка: Создайте базу данных learn и заполните коллекцию unicorns указанными документами.

Команда(ы):

```
use learn

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

 ... (остальные 10 insert)

// 12 – «второй способ»
```

var doc = {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}

db.unicorns.insert(doc)

// 13 – проверка

db.unicorns.find().pretty()

Лог (скриншот):

Практическое задание 2.2.1

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

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

Команда(ы):

```
// самцы

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

// самки, первые три

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

```
learn>  db.unicorns.find({gender:'m'}).sort({name:1})
[
  {
    _id: ObjectId('682ef36c262a32d8cd6c4bdc'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

```
learn>  db.unicorns.find({gender:'f'}).sort({name:1}).limit(3)
[
  {
    _id: ObjectId('682ef2d2262a32d8cd6c4bdb'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

Практическое задание 2.2.2

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

Команда(ы):

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

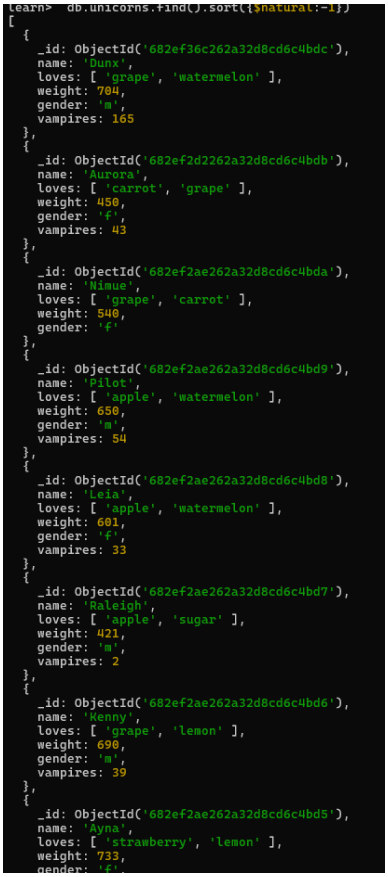
```
learn>  db.unicorns.find({gender:'m'}, {_id:0, loves:0})
[
  { name: 'Horny', weight: 600, gender: 'm', vampires: 63 },
  { name: 'Unicrom', weight: 984, gender: 'm', vampires: 182 },
  { name: 'Roooooodles', weight: 575, gender: 'm', vampires: 99 },
  { name: 'Kenny', weight: 690, gender: 'm', vampires: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

Практическое задание 2.2.3

Формулировка: **Вывести список единорогов в обратном порядке добавления.**

Команда(ы):

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



Практическое задание 2.1.4

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

Команда(ы):
db.unicorns.find({}, {_id:0, name:1, firstLove:{ $slice:['$loves',1] }})

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

Практическое задание 2.3.1

Формулировка: **Вывести список самок единорогов весом от полутонны до 700 кг (без _id).**

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

```
learn>  db.unicorns.find({gender:'f', weight:{$gte:500,$lte:700}}, {_id:0})
[
  {
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.2

Формулировка: **Вывести список самцов единорогов весом ≥ 500 кг, предпочитающих grape и lemon (без _id).**

Команда(ы):

db.unicorns.find({gender:'m', weight:{$gte:500}, loves:{$all:['grape','lemon']}}, {_id:0})

```
learn>  db.unicorns.find({gender:'m', weight:{$gte:500}, loves:{$all:['grape','lemon']}}, {_id:0})
[
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

Практическое задание 2.3.3

Формулировка: **Найти всех единорогов, не имеющих ключ vampires.**

Команда(ы):

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

```
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>  db.unicorns.find({vampires:{$exists:false}})
[
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.4

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

Команда(ы):

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

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

Практическое задание 3.1.1

Формулировка: **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"}}

Сформировать запрос, который возвращает список городов с независимыми мэрами (party="I"). Вывести только название города и информацию о мэре.

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

Команда(ы):

```
// независимые мэры

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

// party отсутствует

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

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

Практическое задание 3.1.2

Формулировка: 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.

Содержание коллекции единорогов unicorns:

Команда(ы):

// 3) функция-фильтр

var male = function () { return this.gender === 'm'; }

// 4-5) курсор (mongosh ≥ 1.11)

var c = db.unicorns.find({ $where: male }).sort({ name: 1 }).limit(2);

c.forEach(printjson);

```
learn> // 3) функция-фильтр
... var male = function () { return this.gender === 'm'; }
...
... // 4-5) курсор (mongosh ≥ 1.11)
... var c = db.unicorns.find({ $where: male }).sort({ name: 1 }).limit(2);
... c.forEach(printjson);
...
{
  _id: ObjectId('682ef36c262a32d8cd6c4bdc'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('682ef2ae262a32d8cd6c4bd0'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

Практическое задание 3.2.1

Формулировка: **Вывести количество самок единорогов весом от полутонны до 600 кг.**

Команда(ы):

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

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

Практическое задание 3.2.2

Формулировка: **Вывести список предпочтений (loves).**

Команда(ы):

```
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

Формулировка: **Посчитать количество единорогов обоих полов.**

Команда(ы):

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

```
learn>  db.unicorns.aggregate([{ $group:{ _id:'$gender', count:{ $sum:1 } } }])
[ { _id: 'm', count: 7 }, { _id: 'f', count: 5 } ]
learn>
```

Практическое задание 3.3.1

Формулировка: **Сохранить документ Barny в коллекции unicorns и проверить содержимое.**

Команда(ы):

```
// добавить Barny

db.unicorns.insertOne({

 name: 'Barny',

 loves: ['grape'],

 weight: 340,

 gender: 'm'

})


// проверка

db.unicorns.find({ name: 'Barny' })
```

```
learn> // добавить Barny
... db.unicorns.insertOne({
...   name: 'Barny',
...   loves: ['grape'],
...   weight: 340,
...   gender: 'm'
... })
...
... // проверка
... db.unicorns.find({ name: 'Barny' })
...
[
  {
    _id: ObjectId('682efa08262a32d8cd6c4be0'),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
```

Практическое задание 3.3.2

Формулировка: **Изменить Ayna: вес = 800, vampires = 51 и проверить коллекцию.**

Команда(ы):

db.unicorns.update({name:'Ayna'}, { $set:{ weight:800, vampires:51 } })

db.unicorns.find({name:'Ayna'})

```
learn>  db.unicorns.update({name:'Ayna'}, { $set:{ weight:800, vampires:51 } })
...  db.unicorns.find({name:'Ayna'})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
[
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

Практическое задание 3.3.3

Формулировка: **Изменить Raleigh: добавить предпочтение redbull и проверить коллекцию.**

Команда(ы):

```
db.unicorns.update({name:'Raleigh'}, { $addToSet:{ loves:'redbull' } })
```

```
db.unicorns.find({name:'Raleigh'})
```

Практическое задание 3.3.4

Формулировка: **Всем самцам увеличить vampires на 5 и проверить коллекцию.**

Команда(ы):

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

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

```
learn>  db.unicorns.update({gender:'m'}, { $inc:{ vampires:5 } }, { multi:true })
...   db.unicorns.find({gender:'m'}, {name:1, vampires:1})
[
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd0'),
    name: 'Horny',
    vampires: 68
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd2'),
    name: 'Unicrom',
    vampires: 187
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd3'),
    name: 'Roooooodles',
    vampires: 104
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd6'),
    name: 'Kenny',
    vampires: 44
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd7'),
    name: 'Raleigh',
    vampires: 7
  },
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd9'),
    name: 'Pilot',
    vampires: 59
  },
  {
    _id: ObjectId('682ef36c262a32d8cd6c4bdc'),
    name: 'Dunx',
    vampires: 170
  },
  {
    _id: ObjectId('682efa08262a32d8cd6c4be0'),
    name: 'Barny',
    vampires: 5
  }
]
```

Практическое задание 3.3.5

Формулировка: **Обновить город Portland: мэр беспартийный и проверить коллекцию towns.**

Команда(ы):

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

```
db.towns.find({name:'Portland'})
```

```
learn>  db.towns.update({name:'Portland'}, { $unset:{ 'mayor.party':1 } })
...   db.towns.find({name:'Portland'})
[
  {
    _id: ObjectId('682ef7e7262a32d8cd6c4bdf'),
    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

Формулировка: **Изменить Pilot: теперь любит chocolate и проверить коллекцию.**

Команда(ы):

db.unicorns.update({name:'Pilot'}, { $addToSet:{ loves:'chocolate' } })

db.unicorns.find({name:'Pilot'})

```
learn> db.unicorns.update({name:'Pilot'}, { $addToSet:{ loves:'chocolate' } })
...   db.unicorns.find({name:'Pilot'})
[
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

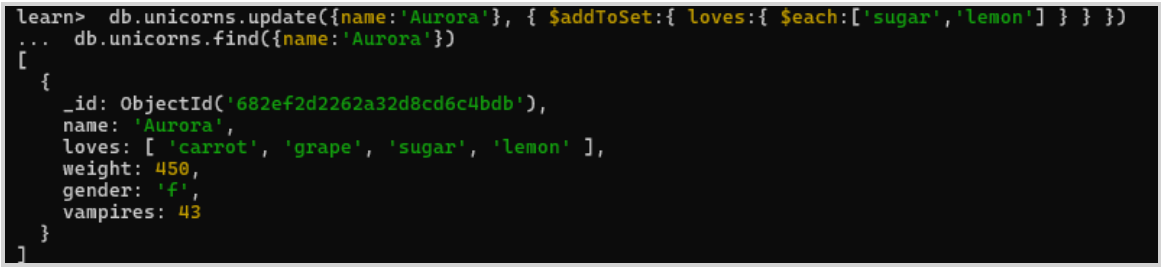
Практическое задание 3.3.7

Формулировка: **Изменить Aurora: теперь любит sugar и lemon и проверить коллекцию.**

Команда(ы):

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

db.unicorns.find({name:'Aurora'})



Практическое задание 3.4.1

Формулировка: 4) Создать коллекцию towns (если не создана) и вставить три документа.5) Удалить документы с беспартийными мэрами.6) Проверить содержимое.7) Очистить коллекцию.8) Просмотреть список коллекций.

Команда(ы):

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

db.towns.find()

db.towns.remove({})

```
show collections
```

```
learn>  db.towns.find()
[
  {
    _id: ObjectId('682efbb4262a32d8cd6c4be1'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  },
  {
    _id: ObjectId('682efbbe262a32d8cd6c4be2'),
    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('682efc42262a32d8cd6c4be4'),
    name: 'Punxsutawney',
    population: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ 'phil the groundhog' ],
    mayor: { name: 'Jim Wehrle' }
  }
]
learn>  db.towns.remove({'mayor.party':{$exists:false}})
{ acknowledged: true, deletedCount: 1 }
learn>  db.towns.find()
[
  {
    _id: ObjectId('682efbb4262a32d8cd6c4be1'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  },
  {
    _id: ObjectId('682efbbe262a32d8cd6c4be2'),
    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' }
  }
]
learn>  db.towns.remove({})
{ acknowledged: true, deletedCount: 2 }
learn> show collections
towns
unicorns
learn>
```

Практическое задание 4.1.1

Формулировка: **Создать коллекцию зон обитания, добавить ссылки у нескольких единорогов, проверить unicorns.**

Команда(ы):

db.habitats.insert({_id:'forest', full:'Enchanted Forest', descr:'Dense magical woods'})

db.unicorns.update({name:'Horny'}, { $set:{ habitat:{ $ref:'habitats', $id:'forest' } } })

db.unicorns.find({habitat:{$exists:true}})

```
learn>  db.habitats.insert({_id:'forest', full:'Enchanted Forest', descr:'Dense magical woods'})
...  db.unicorns.update({name:'Horny'}, { $set:{ habitat:{ $ref:'habitats', $id:'forest' } } })
...  db.unicorns.find({habitat:{$exists:true}})
...
[
  {
    _id: ObjectId('682ef2ae262a32d8cd6c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    habitat: DBRef('habitats', 'forest')
  }
]
```

Практическое задание 4.2.1

Формулировка: **Проверить возможность установки unique-индекса на name в коллекции unicorns.**

Команда(ы):

db.unicorns.createIndex({name:1}, {unique:true})

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

Практическое задание 4.3.1

Формулировка: **Получить все индексы коллекции unicorns, удалить все кроме _id_, попытаться удалить _id_.**

Команда(ы):

```
db.unicorns.getIndexes()

db.unicorns.dropIndexes()

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

```
name_1
learn>  db.unicorns.getIndexes()
...   db.unicorns.dropIndexes()
...   db.unicorns.dropIndex('_id_')
...
MongoServerError[InvalidOptions]: cannot drop _id index
learn>
```

Практическое задание 4.4.1

Формулировка: **Создать объёмную коллекцию numbers, выполнить запрос последних 4 документов, сравнить план без/с индексом на value.**

Команда(ы):

```
// 1

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

// 2

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

// 3

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

// 4

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

// 5

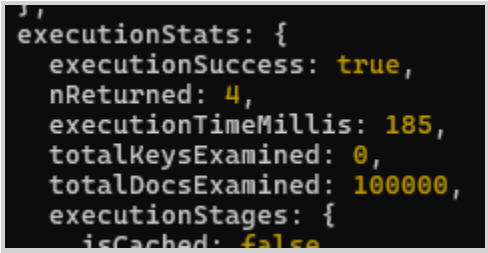
```
db.numbers.getIndexes()
```

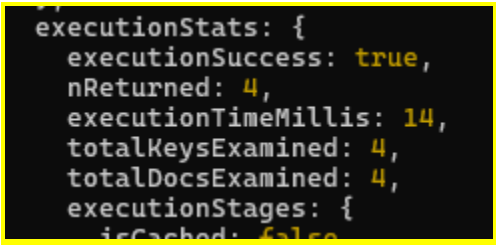
// 6

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

// 7

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





Вывод:

Запрос с индексом выполняется значительно быстрее. Это объясняется тем, что MongoDB использует B-tree индекс, позволяющий мгновенно найти и отсортировать данные по полю value. Без индекса пришлось бы просматривать все документы (full collection scan).

В ходе выполнения лабораторной работы:

1. Созданы и наполнены коллекции *unicorns*, *towns* и *numbers*; освоены команды *insertOne/Many*, *updateOne/Many*, *replaceOne* и *deleteMany*.

2. Реализованы выборки по сложным условиям, включая сравнения, логические операторы и использование $where.

3. Изучены агрегационные операции — группировка, подсчёт, проекция и сортировка — что позволило получить статистику по полу, весу и предпочтениям единорогов.

4. Выполнены массовые изменения данных (операторы $set, $push, $inc), подтвержденные проверочными запросами.

5. Созданы, проанализированы и удалены индексы, а также выявлено влияние уникальных и составных индексов на корректность и скорость запросов.

6. С помощью explain("executionStats") проведён анализ планов выполнения, что на практике доказало рост производительности после добавления нужных индексов.

7. Отчёт сформирован: все команды, скриншоты и краткие выводы внесены в шаблон; код и подписи структурированы и визуально выделены.

Общий итог

Работа дала целостное представление о CRUD-операциях, агрегировании, индексировании и оптимизации запросов в MongoDB. Практические эксперименты подтвердили, что правильный выбор индексов и рациональная схема данных критически влияют на быстродействие системы. Полученные навыки могут быть напрямую

использованы при проектировании и сопровождении реальных документ-ориентированных баз данных.