

**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 Корольков Артем Алексеевич

Факультет прикладной информатики

Группа K3240

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2023

Преподаватель Говорова Марина Михайловна

Санкт-Петербург

2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
ПРАКТИЧЕСКАЯ ЧАСТЬ	4
2 CRUD-ОПЕРАЦИИ В СУБД MONGODB. ВСТАВКА ДАННЫХ. ВЫБОРКА ДАННЫХ	4
2.1 ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ	4
Практическое задание 2.1.1	4
2.2 ВЫБОРКА ДАННЫХ ИЗ БД	6
Практическое задание 2.2.1:	6
Практическое задание 2.2.2:	10
Практическое задание 2.2.3:	11
Практическое задание 2.2.4:	12
2.3 ЛОГИЧЕСКИЕ ОПЕРАТОРЫ	13
Практическое задание 2.3.1:	13
Практическое задание 2.3.2:	14
Практическое задание 2.3.3:	14
Практическое задание 2.3.4:	15
3 ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB. ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ	17
3.1 ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ	17
Практическое задание 3.1.1:	17
Практическое задание 3.1.2:	19
3.2 АГРЕГИРОВАННЫЕ ЗАПРОСЫ	21
Практическое задание 3.2.1:	21
Практическое задание 3.2.2:	22
Практическое задание 3.2.3:	22
3.3 РЕДАКТИРОВАНИЕ ДАННЫХ	23
Практическое задание 3.3.1:	23
Практическое задание 3.3.2:	24
Практическое задание 3.3.3:	25
Практическое задание 3.3.4:	26
Практическое задание 3.3.5:	27
Практическое задание 3.3.6:	28
Практическое задание 3.3.7:	30
3.4 УДАЛЕНИЕ ДАННЫХ ИЗ КОЛЛЕКЦИИ	31
Практическое задание 3.4.1:	31
Контрольные вопросы:	33
4 ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB	36
4.1 ССЫЛКИ В БД	36
Практическое задание 4.1.1:	36
4.2 НАСТРОЙКА ИНДЕКСОВ	38
Практическое задание 4.2.1:	38
4.3 УПРАВЛЕНИЕ ИНДЕКСАМИ	39
Практическое задание 4.3.1:	39
4.4 ПЛАН ЗАПРОСА	40

Практическое задание 4.4.1:	40
Контрольные вопросы:	42
ЗАКЛЮЧЕНИЕ	45
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	46

ВВЕДЕНИЕ

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.

ПРАКТИЧЕСКАЯ ЧАСТЬ

2 CRUD-ОПЕРАЦИИ В СУБД MONGODB. ВСТАВКА ДАННЫХ. ВЫБОРКА ДАННЫХ

2.1 ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ

Практическое задание 2.1.1

Создание БД



Рисунок 1 – Создание БД.

Заполнение коллекции единорогов unicorns:

```

> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('682f4180a566f333e427a23b')
  }
}
> db.unicorns.find()
< {
  _id: ObjectId('682f4180a566f333e427a231'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId('682f4180a566f333e427a232'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}

```

Рисунок 2 – Скриншот заполнения коллекции единорогов unicorns.

Второй способ вставки:

```

> db.unicorns.remove({})
< DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
< {
  acknowledged: true,
  deletedCount: 11
}
> db.unicorns.find()
<
> document = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165});
db.unicorns.insert(document);
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('682f43b6a566f333e427a23c')
  }
}

```

Рисунок 3 – Скриншот второго способа заполнения коллекции единорогов unicorns.

Вывод

```
> db.unicorns.find()
< {
  _id: ObjectId('682f43b6a566f333e427a23c'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
},
{
  _id: ObjectId('682f440da566f333e427a23d'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
},
{
  _id: ObjectId('682f440da566f333e427a23e'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
  _id: ObjectId('682f440da566f333e427a23f'),
  name: 'Unicrom',
```

Рисунок 4 – Скриншот вывода коллекции единорогов unicorns.

2.2 ВЫБОРКА ДАННЫХ ИЗ БД

Практическое задание 2.2.1:

Сформируйте запросы для вывода списков самцов и самок единорогов.

Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.

Все самцы, отсортированные по имени:

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

```
> db.unicorns.find({gender: 'm'}).sort({name: 1})
< {
  _id: ObjectId('682f43b6a566f333e427a23c'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('682f440da566f333e427a23d'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId('682f440da566f333e427a243'),
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
{
  _id: ObjectId('682f440da566f333e427a246'),
  name: 'Difst'
```

Рисунок 5 – Скриншот вывода самцов, отсортированных по имени.

Все самки, отсортированные по имени (первые 3)

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

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
< {
  _id: ObjectId('682f440da566f333e427a23e'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId('682f440da566f333e427a242'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 733,
  gender: 'f',
  vampires: 40
}
{
  _id: ObjectId('682f440da566f333e427a245'),
  name: 'Leia',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 601,
  gender: 'f',
  vampires: 33
}
test> |
```

Рисунок 6 – Скриншот вывода первых 3 самцов, отсортированных по имени.

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

Первая самка, любящая carrot (вариант с findOne)

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

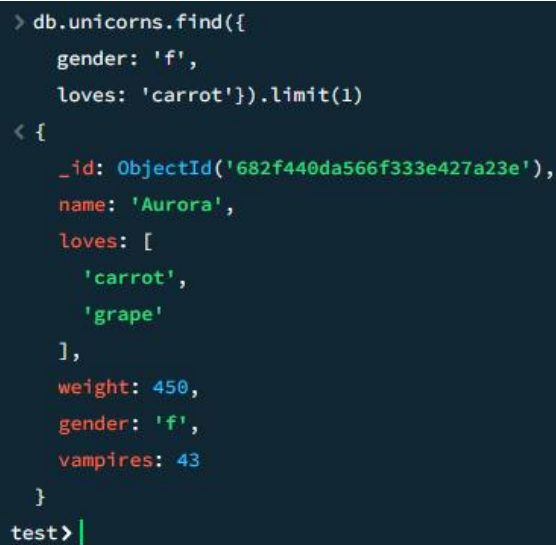


```
> db.unicorns.findOne({  
  gender: 'f',  
  loves: 'carrot'})  
< {  
  _id: ObjectId('682f440da566f333e427a23e'),  
  name: 'Aurora',  
  loves: [  
    'carrot',  
    'grape'  
  ],  
  weight: 450,  
  gender: 'f',  
  vampires: 43  
}
```

Рисунок 7 – Скриншот вывода первой самки, любящей carrot.

Первая самка, любящая carrot (вариант с limit)

```
db.unicorns.find({  
  gender: 'f',  
  loves: 'carrot'}).limit(1)
```



```
> db.unicorns.find({  
  gender: 'f',  
  loves: 'carrot'}).limit(1)  
< {  
  _id: ObjectId('682f440da566f333e427a23e'),  
  name: 'Aurora',  
  loves: [  
    'carrot',  
    'grape'  
  ],  
  weight: 450,  
  gender: 'f',  
  vampires: 43  
}
```

Рисунок 8 – Скриншот вывода первой самки, любящей carrot (вариант с limit).

Практическое задание 2.2.2:

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

Список самцов без информации о предпочтениях и весе

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

```
> db.unicorns.find(  
  {gender: 'm'},  
  {loves: 0, weight: 0, _id: 0})  
< {  
  name: 'Dunx',  
  gender: 'm',  
  vampires: 165  
}  
{  
  name: 'Horny',  
  gender: 'm',  
  vampires: 63  
}  
{  
  name: 'Unicrom',  
  gender: 'm',  
  vampires: 182  
}  
{  
  name: 'Rooooooodles',  
  gender: 'm',  
  vampires: 99  
}  
{  
  name: 'Kenny',  
  gender: 'm',  
  vampires: 39  
}
```

Рисунок 9 – Скриншот вывода самцов без информации о предпочтениях и весе.

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

Список единорогов в обратном порядке добавления

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

```
> db.unicorns.find().sort({$natural: -1})
< {
  _id: ObjectId('682f440da566f333e427a247'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  _id: ObjectId('682f440da566f333e427a246'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 650,
  gender: 'm',
  vampires: 54
}
{
  _id: ObjectId('682f440da566f333e427a245'),
  name: 'Leia',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 601,
  gender: 'f',
  vampires: 33
}
{
  _id: ObjectId('682f440da566f333e427a244'),
  name: 'Raleigh',
  loves: [
```

Рисунок 9 – Скриншот вывода списка единорогов в обратном порядке добавления.

Практическое задание 2.2.4:

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

Список единорогов с первым любимым предпочтением (без `_id`)

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

Вариант с явным указанием всех нужных полей

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



```
> db.unicorns.find(  
  {},  
  {loves: {$slice: 1}, _id: 0})  
< {  
  name: 'Dunx',  
  loves: [  
    'grape'  
  ],  
  weight: 704,  
  gender: 'm',  
  vampires: 165  
}  
{  
  name: 'Horny',  
  loves: [  
    'carrot'  
  ],  
  weight: 600,  
  gender: 'm',  
  vampires: 63  
}
```

Рисунок 10 – Скриншот вывода списка единорогов с названием первого любимого предпочтения, исключив идентификатор.

2.3 ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Практическое задание 2.3.1:

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

Самки весом от 500 до 700 кг (без _id)

```
db.unicorns.find(  
  {  
    gender: 'f',  
    weight: {$gte: 500, $lte: 700}  
  },  
  {_id: 0}).sort({weight: 1})
```

```
> 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'  
}  
test>
```

Рисунок 11 – Скриншот вывода списка самок единорогов весом от полутонны до 700 кг, исключив вывод идентификатора.

Практическое задание 2.3.2:

Вывести список самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора.

Самцы весом ≥ 500 кг, любящие grape И lemon (без _id)

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



```
> 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  
}
```

Рисунок 12 – Скриншот вывода списка самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора.

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

Единороги без поля vampires

```
> db.unicorns.find(
  {vampires: {$exists: false}})
< {
  _id: ObjectId('682f440da566f333e427a247'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
```

Рисунок 13 – Скриншот вывода списка всех единорогов, не имеющих ключ vampires.

Практическое задание 2.3.4:

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

Имена самцов с первым предпочтением (отсортировано по имени)

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

```
> db.unicorns.find(
  {gender: 'm'},
  {
    name: 1,
    loves: {$slice: 1},
    _id: 0
  }).sort({name: 1})
< {
  name: 'Dunx',
  loves: [
    'grape'
  ]
}
{
  name: 'Horny',
  loves: [
    'carrot'
  ]
}
{
  name: 'Kenny',
  loves: [
    'grape'
  ]
}
```

Рисунок 14 – Скриншот вывода упорядоченного списка имен самцов единорогов с информацией об их первом предпочтении.

3 ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB.

ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ

3.1 ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ

Практическое задание 3.1.1:

1. Создайте коллекцию towns, включающую следующие документы:

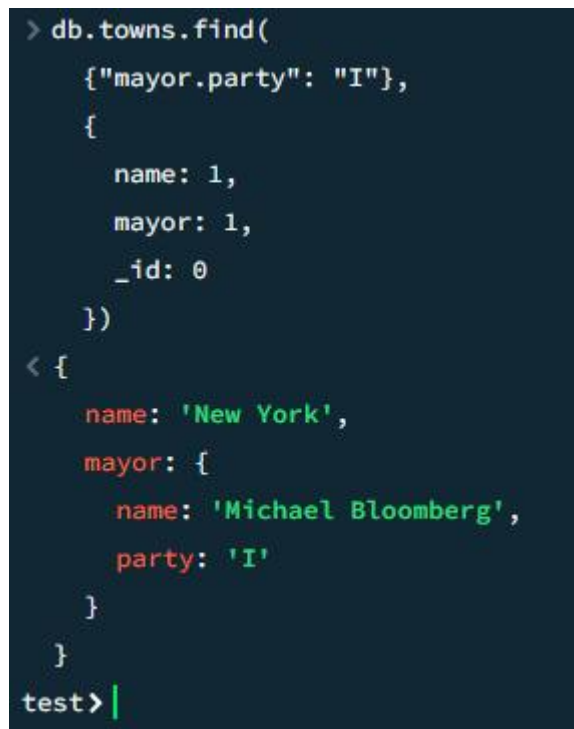
```
> db.towns.insertMany([
  {
    name: "Punxsutawney",
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31"),
    famous_for: [""],
    mayor: {
      name: "Jim Wehrle"
    }
  },
  {
    name: "New York",
    populatiuon: 22200000,
    last_sensus: ISODate("2009-07-31"),
    famous_for: ["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('68304d01445f6b1398f8139e'),
    '1': ObjectId('68304d01445f6b1398f8139f'),
    '2': ObjectId('68304d01445f6b1398f813a0')
  }
}
```

Рисунок 15 – Скриншот создания коллекции towns.

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

Города с независимыми мэрами (party="I")

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



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

Рисунок 16 – Скриншот запроса, который возвращает список городов с независимыми мэрами (party="I").

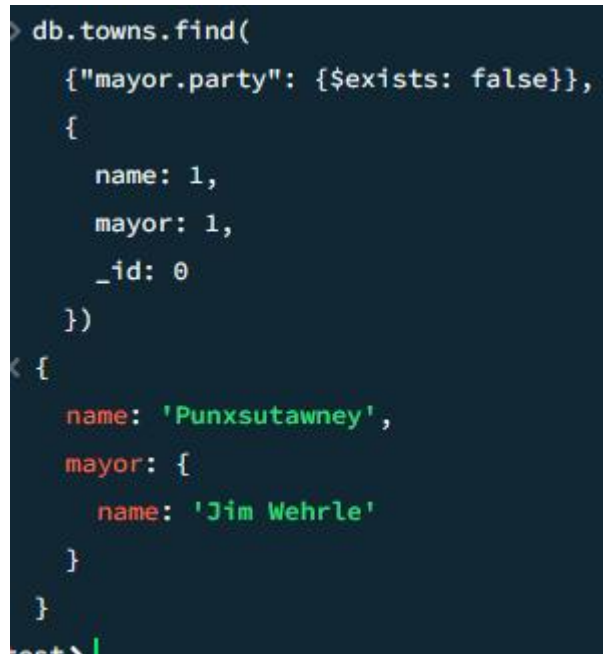
3. Сформировать запрос, который возвращает список беспартийных мэров (party отсутствует). Вывести только название города и информацию о мэре.

Города с беспартийными мэрами (отсутствует поле party)

```

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

```



```

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

```

Рисунок 17 – Скриншот запроса, который возвращает список беспартийных мэров (party отсутствует).

Практическое задание 3.1.2:

4. Сформировать функцию для вывода списка самцов единорогов.

Функция для фильтрации самцов

```

function findMaleUnicorns() {
  return db.unicorns.find({gender: 'м'});}

```

```

> function findMaleUnicorns() {
    return db.unicorns.find({gender: 'm'});}
< [Function: findMaleUnicorns]
> findMaleUnicorns()
< [
  {
    _id: ObjectId('682f43b6a566f333e427a23c'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('682f440da566f333e427a23d'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
]

```

Рисунок 18 – Скриншот функции для вывода списка самцов единорогов.

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

Создаём курсор с сортировкой и ограничением

```


var cursor = db.unicorns.find({gender: 'm'})
    .sort({name: 1})
    .limit(2);

```

6. Вывести результат, используя `forEach`.

Обработка курсора с помощью `forEach`

```
cursor.forEach(function(unicorn) {  
  print("И м я : " + unicorn.name +  
    ", В е с : " + unicorn.weight +  
    ", Л ю б и т : " + unicorn.loves.join(', '));});
```



```
> var cursor = db.unicorns.find({gender: 'm'})  
  .sort({name: 1})  
  .limit(2);  
  
> cursor.forEach(function(unicorn) {  
  print("Имя: " + unicorn.name +  
    ", Вес: " + unicorn.weight +  
    ", Любит: " + unicorn.loves.join(', '));  
});  
  
< Имя: Dunx, Вес: 704, Любит: grape, watermelon  
< Имя: Horny, Вес: 600, Любит: carrot, papaya  
test>
```

Рисунок 19 – Скриншот обработки курсора с помощью `forEach`.

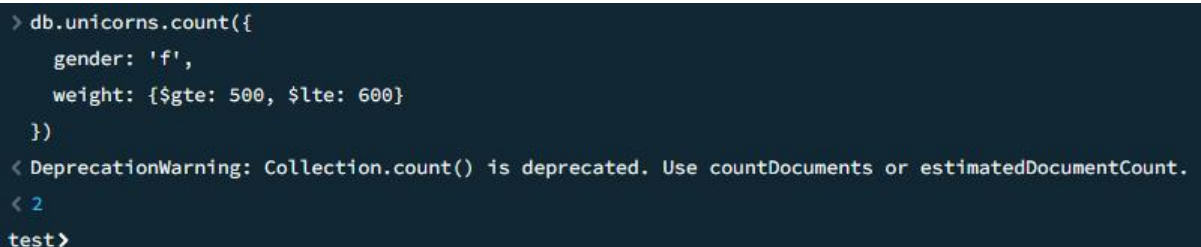
3.2 АГРЕГИРОВАННЫЕ ЗАПРОСЫ

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

Количество самок весом от 500 до 600 кг

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



```
> db.unicorns.count({  
  gender: 'f',  
  weight: {$gte: 500, $lte: 600}  
})  
  
< DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.  
< 2  
test>
```

Рисунок 20 – Скриншот вывода количества самок единорогов весом от полутонны до 600 кг.

Практическое задание 3.2.2:

Вывести список предпочтений.

Список всех уникальных предпочтений

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



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

Рисунок 21 – Скриншот вывода списка всех уникальных предпочтений.

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

Количество особей по полу

```
db.unicorns.aggregate([
  {$group: {
    _id: "$gender",
    count: {$sum: 1},
    avgWeight: {$avg: "$weight"},
    totalVampires: {$sum: "$vampires"}
  }}})
```

```

> db.unicorns.aggregate([
  {$group: {
    _id: "$gender",
    count: {$sum: 1},
    avgWeight: {$avg: "$weight"},
    totalVampires: {$sum: "$vampires"}
  }}})
< {
  _id: 'm',
  count: 7,
  avgWeight: 660.5714285714286,
  totalVampires: 604
}
{
  _id: 'f',
  count: 5,
  avgWeight: 574.8,
  totalVampires: 196
}
test>

```

Рисунок 22 – Скриншот вывода количества особей единорогов обоих полов.

3.3 РЕДАКТИРОВАНИЕ ДАННЫХ

Практическое задание 3.3.1:

1. Выполнить команду:

```

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

```

```

> db.unicorns.insertOne({
  name: 'Barney',
  loves: ['grape'],
  weight: 340,
  gender: 'm'
})
< {
  acknowledged: true,
  insertedId: ObjectId('68305282445f6b1398f813a1')
}
test>

```

Рисунок 23 – Скриншот выполнения команды.

2. Проверить содержимое коллекции unicorns.

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



```
> db.unicorns.find({name: 'Barney'}).pretty()
< {
  _id: ObjectId('68305282445f6b1398f813a1'),
  name: 'Barney',
  loves: [
    'grape'
  ],
  weight: 340,
  gender: 'm'
}
test> |
```

Рисунок 24 – Скриншот проверки содержимого коллекции unicorns.

Практическое задание 3.3.2:

1. Для самки единорога Айна внести изменения в БД: теперь её вес 800, она убила 51 вампира. Set - частичное обновление.

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

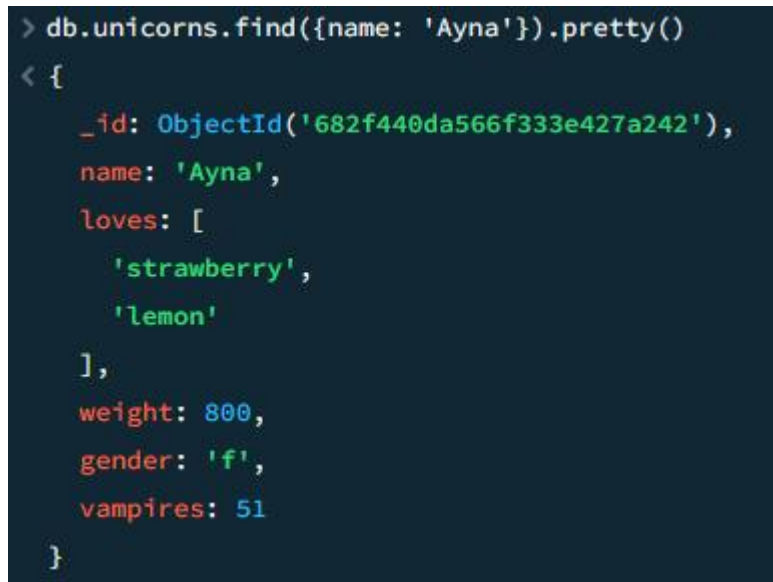


```
> db.unicorns.updateOne(
  {name: 'Ayna'},
  {
    $set: {
      weight: 800,
      vampires: 51
    }
  }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Рисунок 25 – Скриншот внесения изменения в БД для самки единорога Айна.

2. Проверить содержимое коллекции unicorns.

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




```
> db.unicorns.find({name: 'Ayna'}).pretty()
< {
  _id: ObjectId('682f440da566f333e427a242'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 800,
  gender: 'f',
  vampires: 51
}
```

Рисунок 26 – Скриншот проверки содержимого.

Практическое задание 3.3.3:

1. Для самца единорога Raleigh внести изменения в БД: теперь он любит рэдбул.

```
db.unicorns.update(
  {name: 'Raleigh'},
  {$set: {loves: ['apple', 'sugar', 'redbull']}})
```

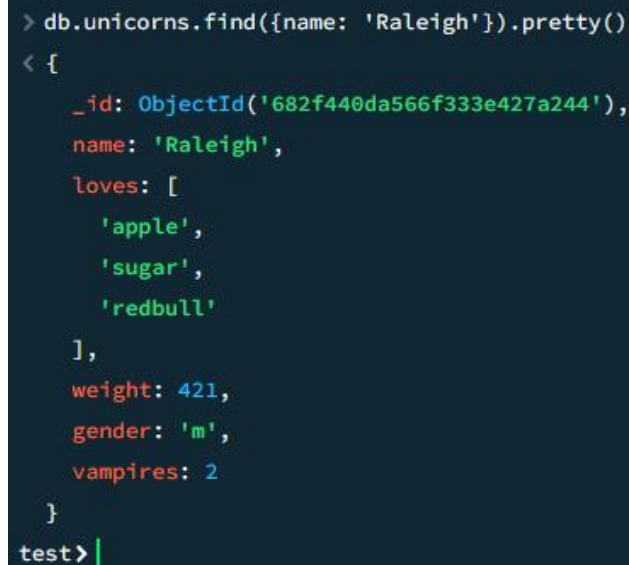


```
> db.unicorns.update(
  {name: 'Raleigh'},
  {$set: {loves: ['apple', 'sugar', 'redbull']}}
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> |
```

Рисунок 27 – Скриншот внесения изменения в БД для самца единорога Raleigh.

2. Проверить содержимое коллекции unicorns.

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



```
> db.unicorns.find({name: 'Raleigh'}).pretty()
< {
  _id: ObjectId('682f440da566f333e427a244'),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar',
    'redbull'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
```

Рисунок 28 – Скриншот проверки содержимого коллекции unicorns.

Практическое задание 3.3.4:

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.

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



```
> db.unicorns.updateMany(
  {gender: 'm', vampires: {$exists: true}},
  {$inc: {vampires: 5}}
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
```

Рисунок 29 – Скриншот увеличения количества убитых вампиров на 5 для всех самцов.

2. Проверить содержимое коллекции unicorns.

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



```
> db.unicorns.find(  
  {gender: 'm'},  
  {name: 1, vampires: 1, _id: 0}  
).pretty()  
< {  
  name: 'Dunx',  
  vampires: 170  
}  
{  
  name: 'Horny',  
  vampires: 68  
}  
{  
  name: 'Unicrom',  
  vampires: 187  
}  
{  
  name: 'Roooooodles',  
  vampires: 104  
}  
{  
  name: 'Kenny',  
  vampires: 44  
}
```

Рисунок 30 – Скриншот проверки содержимого коллекции unicorns.

Практическое задание 3.3.5:

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

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

```

> db.towns.update(
  {name: 'Portland'},
  {$unset: {'mayor.party': 1}}
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test>

```

Рисунок 31 – Скриншот изменения информации о городе Портланд.

2. Проверить содержимое коллекции towns.

```

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

```

```

> db.towns.find(
  {name: 'Portland'},
  {name: 1, mayor: 1, _id: 0}
).pretty()
< {
  name: 'Portland',
  mayor: {
    name: 'Sam Adams'
  }
}
test>

```

Рисунок 32 – Скриншот проверки содержимого коллекции unicorns.

Практическое задание 3.3.6:

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

```

db.unicorns.update(
  {name: 'Pilot'},

```

```
{ $push: { loves: 'chocolate' } })
```

```
> db.unicorns.update(
  {name: 'Pilot'},
  { $push: { loves: 'chocolate' } }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test>
```

Рисунок 33 – Скриншот изменения информации о самце единорога Pilot.

2. Проверить содержимое коллекции unicorns.

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


```
> db.unicorns.find(
  {name: 'Pilot'},
  {name: 1, loves: 1, _id: 0}
).pretty()
< {
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ]
}
test> |
```

Рисунок 34 – Скриншот проверки содержимого коллекции unicorns.

Практическое задание 3.3.7:

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

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



```
> db.unicorns.update(  
  {name: 'Aurora'},  
  {$addToSet: {loves: {$each: ['sugar', 'lemon']}}}  
)  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}  
test>
```

Рисунок 35 – Скриншот изменения информации о самке единорога Aurora.

2. Проверить содержимое коллекции unicorns.

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



```
> db.unicorns.find(  
  {name: 'Aurora'},  
  {name: 1, loves: 1, _id: 0}  
).pretty()  
< {  
  name: 'Aurora',  
  loves: [  
    'carrot',  
    'grape',  
    'sugar',  
    'lemon'  
  ]  
}  
test>
```

Рисунок 36 – Скриншот проверки содержимого коллекции unicorns.

3.4 УДАЛЕНИЕ ДАННЫХ ИЗ КОЛЛЕКЦИИ

Практическое задание 3.4.1:

1. Создайте коллекцию towns, включающую следующие документы:

```
> db.towns.insertMany([
  {
    name: "Punxsutawney",
    population: 6200,
    last_sensus: ISODate("2008-01-31"),
    famous_for: ["phil the groundhog"],
    mayor: {
      name: "Jim Wehrle"
    }
  },
  {
    name: "New York",
    population: 22200000,
    last_sensus: ISODate("2009-07-31"),
    famous_for: ["status of liberty", "food"],
    mayor: {
      name: "Michael Bloomberg",
      party: "I"
    }
  },
  {
    name: "Portland",
    population: 528000,
    last_sensus: ISODate("2009-07-20"),
    famous_for: ["beer", "food"],
    mayor: {
      name: "Sam Adams",
      party: "D"
    }
  }
])
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68315f4713ae45d2fccb21ec'),
    '1': ObjectId('68315f4713ae45d2fccb21ed'),
    '2': ObjectId('68315f4713ae45d2fccb21ee')
  }
}
```

Рисунок 37 – Скриншот создания коллекции towns.

2. Удалите документы с беспартийными мэрами.

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



```
> db.towns.deleteMany({  
  "mayor.party": {$exists: false}  
})  
< {  
  acknowledged: true,  
  deletedCount: 3  
}  
test>
```

Рисунок 38 – Скриншот удаления документов с беспартийными мэрами.

3. Проверьте содержание коллекции.

```
db.towns.find().pretty()
```



```
> db.towns.find().pretty()  
< {  
  _id: ObjectId('68304d01445f6b1398f8139f'),  
  name: 'New York',  
  populatiuon: 22200000,  
  last_sensus: 2009-07-31T00:00:00.000Z,  
  famous_for: [  
    'status of liberty',  
    'food'  
  ],  
  mayor: {  
    name: 'Michael Bloomberg',  
    party: 'I'  
  }  
}  
{  
  _id: ObjectId('68315f4713ae45d2fccb21ed'),  
  name: 'New York',  
  popujatiuon: 22200000,  
  last_sensus: 2009-07-31T00:00:00.000Z,  
  famous_for: [  
    'status of liberty',  
    'food'  
  ],  
  mayor: {  
    name: 'Michael Bloomberg',  
    party: 'I'  
  }  
}
```

Рисунок 40 – Скриншот проверки содержимого коллекции.

4. Очистите коллекцию.

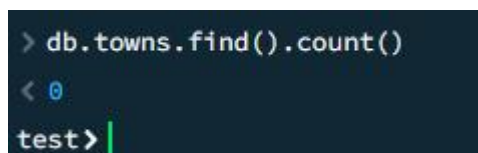
```
db.towns.deleteMany({})
```



```
> db.towns.deleteMany({})  
< {  
  acknowledged: true,  
  deletedCount: 3  
}  
test>
```

Рисунок 41 – Скриншот очистки коллекции.

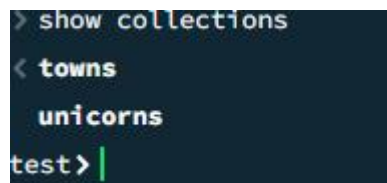
```
db.towns.find().count()
```



```
> db.towns.find().count()  
< 0  
test> |
```

Рисунок 42 – Скриншот проверки очистки коллекции.

5. Просмотрите список доступных коллекций.



```
> show collections  
< towns  
  unicorns  
test> |
```

Рисунок 43 – Скриншот вывода доступных коллекций.

Контрольные вопросы:

1. Как используется оператор точка?

- Оператор точка (.) в MongoDB используется для:
- Доступа к полям вложенных документов: "mayor.name"
- Использования операторов обновления: \$set, \$inc
- Доступа к методам коллекций: db.collection.find()
- Работы с массивами: \$push, \$pull

Примеры:

Доступ к вложенному полю:

```
db.towns.find({"mayor.party": "I"})
```

Обновление вложенного поля

```
db.users.update({}, {$set: {"address.city": "Moscow"}})
```

2. Как можно использовать курсор?

Это специальный объект, который представляет собой указатель на набор результатов запроса к базе данных. Он позволяет итерироваться (перебирать) по документам, полученным в результате выполнения таких операций, как `find()`, `aggregate()` и других.

- Итерироваться по результатам запроса
- Ограничивать количество возвращаемых документов: `limit()`
- Пропускать документы: `skip()`
- Сортировать результаты: `sort()`
- Преобразовывать в массив: `toArray()`
- Применять функции к каждому документу `forEach()`

Примеры:

Создание курсора:

```
const cursor = db.unicorns.find({gender: 'm'}).sort({weight: -1}).limit(5)
```

Итерация по курсору:

```
while (cursor.hasNext()) {  
  printjson(cursor.next())  
}
```

Преобразование в массив:

```
const heavyUnicorns = cursor.toArray()
```

3. Какие возможности агрегирования данных существуют в MongoDB?

Агрегирование данных — это процесс обработки и преобразования данных для получения сводной информации, статистики или аналитических результатов. В MongoDB агрегация выполняется с помощью агрегационного конвейера (Aggregation Pipeline), который состоит из последовательных этапов (стадий), каждый из которых обрабатывает данные и передает их следующей стадии.

- Конвейер агрегации (`$match`, `$group`, `$sort`, `$limit`, `$project`)
- Группировка данных (`$group`)
- Фильтрация (`$match`)

- Преобразование документов (\$project)
- Объединение коллекций (\$lookup)
- Работа с массивами (\$unwind, \$push, \$addToSet)
- Математические операции (\$sum, \$avg, \$max, \$min)
- Текстовые операции (\$concat, \$substr)
- Операторы даты (\$year, \$month, \$dayOfMonth)

4. Какая из функций save или update более детально позволит настроить редактирование документов коллекции?

Метод update() (и его современные аналоги updateOne(), updateMany()) предоставляет более детальный контроль, так как:

- Позволяет использовать операторы обновления (\$set, \$inc, \$push и др.)
- Поддерживает точечную нотацию для вложенных полей
- Позволяет обновлять только определенные поля, а не весь документ
- Имеет параметры upsert и multi для тонкой настройки поведения
- В современных версиях MongoDB метод save() удален, вместо него рекомендуется использовать insertOne() или replaceOne()

5. Как происходит удаление документов из коллекции по умолчанию?

По умолчанию (без параметров) методы удаления работают так:

- deleteMany({}) - удаляет ВСЕ документы в коллекции.
- deleteMany({условие}) - удаляет ВСЕ документы, соответствующие условию.
- deleteOne({условие}) - удаляет ПЕРВЫЙ найденный документ, соответствующий условию.

Важные особенности:

1. Операции удаления атомарны на уровне одного документа.
2. Удаление не освобождает место на диске (для этого нужно выполнить compact).
3. Для безопасного удаления рекомендуется сначала выполнить find() с теми же условиями.

4 ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

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.

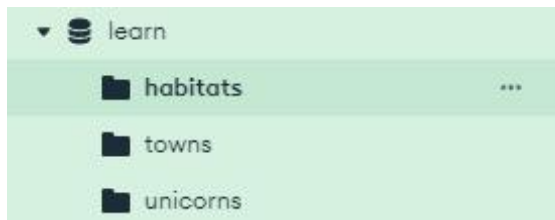


Рисунок 44 – Скриншот создания коллекции habitats.

```
> db.habitats.insertMany([
  {
    _id: "forest",
    name: "Волшебный лес",
    description: "Густой лес с древними деревьями и светящимися грибами"
  },
  {
    _id: "mountain",
    name: "Хрустальные горы",
    description: "Высокие горы с кристальными пещерами"
  },
  {
    _id: "meadow",
    name: "Радужные луга",
    description: "Просторные луга с волшебными цветами"
  }
])
< {
  acknowledged: true,
  insertedIds: {
    '0': 'forest',
    '1': 'mountain',
    '2': 'meadow'
  }
}
```

Рисунок 45 – Скриншот заполнения коллекции habitats.

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

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

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

db.unicorns.updateOne(
  {name: "Unicrom"},
  {$set: {habitat: {$ref: "habitats", $id: "mountain"}}}
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test>
```

Рисунок 46 – Скриншот включения ссылки на зону обитания.

3. Проверьте содержание коллекции единорогов.

```
> db.unicorns.find({habitat: {$exists: true}}).pretty()
< {
  _id: ObjectId('682f440da566f333e427a23d'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 68,
  habitat: DBRef('habitats', 'forest')
}
{
  _id: ObjectId('682f440da566f333e427a23e'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape',
    'sugar',
    'lemon'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43,
  habitat: DBRef('habitats', 'meadow')
}
{
  _id: ObjectId('682f440da566f333e427a23f'),
  name: 'Unicrom',
  loves: [
    'energon',
    'redbull'
  ],
  weight: 984,
  gender: 'm',
  vampires: 187,
  habitat: DBRef('habitats', 'mountain')
}
```

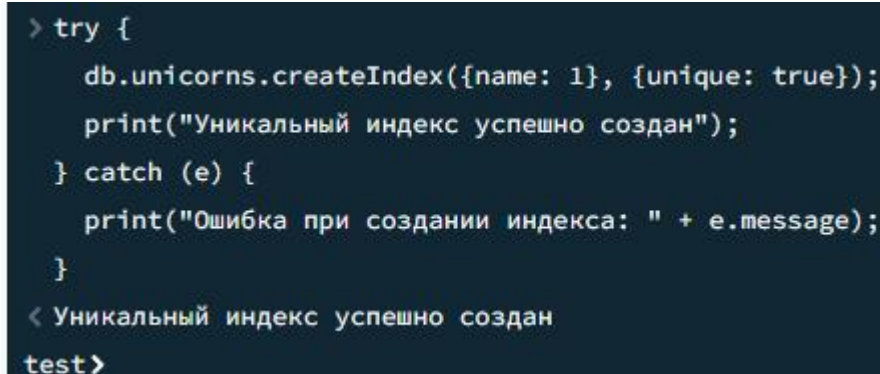
Рисунок 47 – Скриншот проверки содержания коллекции habitats.

4.2 НАСТРОЙКА ИНДЕКСОВ

Практическое задание 4.2.1:

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

```
try {  
  db.unicorns.createIndex({name: 1}, {unique: true});  
  print("Уникальный индекс успешно создан");  
} catch (e) {  
  print("Ошибка при создании индекса: " +  
    e.message);  
}
```



```
> try {  
  db.unicorns.createIndex({name: 1}, {unique: true});  
  print("Уникальный индекс успешно создан");  
} catch (e) {  
  print("Ошибка при создании индекса: " + e.message);  
}  
< Уникальный индекс успешно создан  
test>
```

Рисунок 48 – Скриншот проверки задания для коллекции unicorns индекс для ключа name с флагом unique.

Проверка существующих индексов



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

Рисунок 49 – Скриншот проверки существующих индексов.

Если в коллекции уже есть документы с дублирующимися именами, MongoDB не позволит создать уникальный индекс. В нашем случае все имена уникальны, поэтому индекс создан успешно.

4.3 УПРАВЛЕНИЕ ИНДЕКСАМИ

Практическое задание 4.3.1:

1. Получите информацию о всех индексах коллекции unicorns .

```
db.unicorns.getIndexes()
```



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

Рисунок 50 – Скриншот получения информации о всех индексах коллекции unicorns.

2. Удалите все индексы, кроме индекса для идентификатора.

```
indexes.forEach(function(index) {
  if (index.name !== "_id_") {
    print("У д а л я ю  и н д е к с : " + index.name);
    db.unicorns.dropIndex(index.name);
  });
});
```



```
> const indexes = db.unicorns.getIndexes();

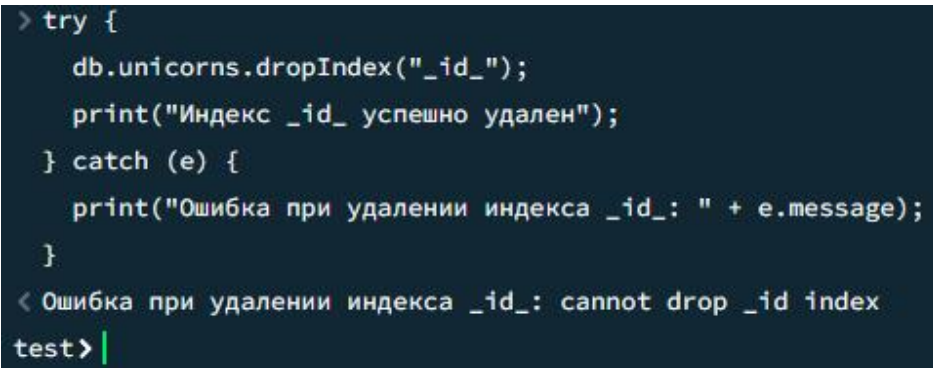
indexes.forEach(function(index) {
  if (index.name !== "_id_") {
    print("Удаляю индекс: " + index.name);
    db.unicorns.dropIndex(index.name);
  }
});

db.unicorns.getIndexes()
< Удаляю индекс: name_1
< [ { v: 2, key: { _id: 1 }, name: '_id_' } ]
test>
```

Рисунок 51 – Скриншот удаления всех индексов, кроме индекса для идентификатора.

3. Попробуйте удалить индекс для идентификатора.

```
try {  
  db.unicorns.dropIndex("_id_");  
  print("Индекс _id_ успешно удален");} catch (e) {  
  print("Ошибка при удалении индекса _id_: " +  
    e.message);}
```



```
> try {  
  db.unicorns.dropIndex("_id_");  
  print("Индекс _id_ успешно удален");  
} catch (e) {  
  print("Ошибка при удалении индекса _id_: " + e.message);  
}  
< Ошибка при удалении индекса _id_: cannot drop _id index  
test> |
```

Рисунок 52 – Скриншот удаления индекса для идентификатора.

Индекс `_id_` является системным и обязательным для каждой коллекции. MongoDB физически не позволяет его удалить, так как этот индекс обеспечивает уникальность поля `_id` — главного идентификатора документа.

4.4 ПЛАН ЗАПРОСА

Практическое задание 4.4.1:

1. Создайте объёмную коллекцию `numbers`, задействовав курсор:

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



```
> for(i = 0; i < 100000; i++) {  
  db.numbers.insert({value: i});  
}  
< DeprecationWarning: Collection.insert() is deprecated. Use  
< {  
  acknowledged: true,  
  insertedIds: {  
    '0': ObjectId('68317029587ae46a0ec9cb7e')  
  }  
}  
learn>
```

Рисунок 53 – Скриншот создания объёмной коллекции `numbers`.


```

> db.numbers.count()
< DeprecationWarning: Collection.count() is
< 100000
learn> |

```

Рисунок 54 – Скриншот проверки созданной коллекции numbers.

2. Выберите последних четыре документа.

```

const query1 = db.numbers.find().sort({value: -1}).limit(4);
query1.forEach(printjson);

```

```

> const query1 = db.numbers.find().sort({value: -1}).limit(4);
  query1.forEach(printjson);
< { _id: ObjectId('68317029587ae46a0ec9cb7e'), value: 99999 }
< { _id: ObjectId('68317029587ae46a0ec9cb7d'), value: 99998 }
< { _id: ObjectId('68317029587ae46a0ec9cb7c'), value: 99997 }
< { _id: ObjectId('68317029587ae46a0ec9cb7b'), value: 99996 }
learn> |

```

Рисунок 55 – Скриншот выбора последних четырех документов.

3. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра executionTimeMillis)

```

const explainWithoutIndex = db.numbers.explain("executionStats")
  .find({value: {$gt: 99995}})
  .sort({value: -1});
printjson(explainWithoutIndex);

```

```

  executionTimeMillis: 54,

```

Рисунок 56 – Скриншот времени выполнения запроса.

4. Создайте индекс для ключа value.

```

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

```

```

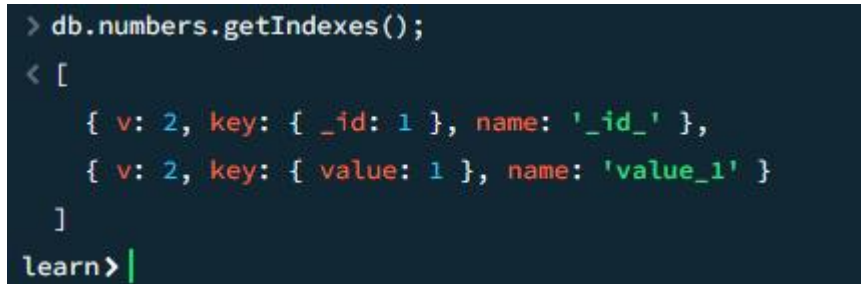
> db.numbers.createIndex({value: 1});
< value_1

```

Рисунок 57 – Скриншот создания индекса для ключа value.

5. Получите информацию о всех индексах коллекции `numbers`.

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



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

Рисунок 58 – Скриншот получения информации о всех индексах коллекции `numbers`.

6. Выполните запрос 2.

```
const explainWithIndex = db.numbers.explain("executionStats")
  .find({value: {$gt: 99995}})
  .sort({value: -1});
printjson(explainWithIndex);
```



```
executionTimeMillis: 7,
```

Рисунок 59 – Скриншот времени выполнения запроса 2.

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

Без индекса потребовалось 54 мс (COLLSCAN - полное сканирование коллекции). С индексом потребовалось 7 мс (IXSCAN - использование индекса)

8. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

Эффективнее запрос с индексом, он работает в 7-8 раз быстрее.

Контрольные вопросы:

1. Назовите способы связывания коллекций в MongoDB.

a) Ручные ссылки (Manual References): В документе хранится только `_id` связанного документа

```
{ user_id: ObjectId("507f1f77bcf86cd799439011") }
```

b) DBRef (Database References)

```
{
  $ref: "collection_name",
  $id: ObjectId("..."),
  $db: "db_name" // опционально
}
```

с) Вложенные документы (Embedded Documents)

```
{
  name: "John",
  address: {
    city: "New York",
    street: "Broadway"
  }
}
```

д) \$lookup (агрегационный pipeline):

```
db.orders.aggregate([
  {
    $lookup: {
      from: "users",
      localField: "user_id",
      foreignField: "_id",
      as: "user_info"
    }
  }
])
```

2. Сколько индексов можно установить на одну коллекцию в БД MongoDB?

Максимально можно установить 64 индекса на одну коллекцию. Также есть ограничение по памяти. Общий размер индексов должен позволять работать в RAM. Обычно 3-5 тщательно подобранных индексов достаточно для большинства коллекций.

3. Как получить информацию о всех индексах базы данных MongoDB?

а) Для конкретной коллекции:

```
db.collection.getIndexes()
```

б) Для всех коллекций в базе:

Получаем список всех коллекций

```
const collections = db.getCollectionNames();
```

/Для каждой коллекции получаем индексы

```
collections.forEach(function(collection) {  
    print("Indexes for " + collection + ":");  
    printjson(db[collection].getIndexes());  
});
```

с) Через системную коллекцию

```
db.system.indexes.find()
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были успешно освоены ключевые аспекты работы с MongoDB, что позволило закрепить практические навыки управления данными в NoSQL-среде. На примере коллекций единорогов и городов отработаны базовые CRUD-операции, включая вставку документов через `insertOne` и `insertMany`, обновление с использованием операторов `$set` и `$inc`, а также удаление данных с применением методов `deleteOne` и `deleteMany`. Особое внимание было уделено работе с вложенными объектами и массивами, где на практике проверены возможности операторов `$push`, `$pull` и `$addToSet` для управления сложными структурами данных.

Эксперименты с агрегационными pipeline продемонстрировали мощь MongoDB для аналитической обработки данных, включая группировку через `$group`, фильтрацию с `$match` и соединение коллекций посредством `$lookup`. Практика с ручными ссылками и `DBRef` раскрыла механизмы связывания документов между коллекциями, что особенно важно для построения реляционных схем в документоориентированной СУБД.

Значительная часть работы была посвящена исследованию индексов, где на конкретных примерах доказана их критическая важность для производительности. Сравнение времени выполнения запросов с индексом (IXSCAN) и без (COLLSCAN) на 100 000 документах наглядно показало, что правильное индексирование может ускорять операции в десятки раз. Анализ планов запросов через `explain()` позволил выработать понимание оптимизации поисковых операций.

Полученный опыт работы с системными ограничениями, такими как невозможность удаления индекса `id`, и обработкой предупреждений об устаревших методах (замена `insert` на `insertMany`) способствовал формированию навыков профессионального администрирования MongoDB. В результате лабораторной работы сформирована комплексная компетенция по управлению данными в MongoDB, включая

проектирование структур, эффективный поиск и модификацию, а также оптимизацию производительности через индексы и агрегационные pipeline.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. MongoDB CRUD Operations [Электронный ресурс] // mongoDB. Documentation: официальный сайт MongoDB. URL: <https://docs.mongodb.com/manual/> (дата обращения: 02.05.2023)
2. MongoDB – Краткое руководство [Электронный ресурс] // CoderLessons.com. Уроки по программированию, DevOps и другим IT-технологиям: сайт, 2019. URL: <https://coderlessons.com/tutorials/bazy-dannykh/uchitsia-mongodb/mongodb-kratkoe-rukovodstvo> (дата обращения: 02.05.2023).
3. Кайл Б. MongoDB в действии [Электронный ресурс] // Доступ в ЭБС «Лань». Режим доступа: <https://e.lanbook.com/book/4156> (дата обращения: 05.05.2025).
4. Онлайн-руководство по MongoDB [Электронный ресурс] // METANIT.COM. Сайт о программировании. URL: <https://metanit.com/nosql/mongodb/> (дата обращения: 05.05.2025).
5. Эрик Р. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. [Электронный ресурс]/Р. Эрик, Р.У. Джим. Электрон. дан. М.: ДМК Пресс, 2013. 384с. Доступ из ЭБС «Лань». URL: <http://e.lanbook.com/book/58690> (дата обращения: 05.05.2025).