**Министерство науки и высшего образования Российской Федерации**
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# Отчет о лабораторной работе 5

«Работа с БД в СУБД MongoDB»

по дисциплине «**Проектирование и реализация баз данных**»

Автор: Королева Екатерина Максимовна

Факультет: ИКТ

Группа: K32422

Преподаватель: Говорова М. М.

Дата: 28.06.2023

Санкт-Петербург 2023

**Цель:** овладеть практическими навыками работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.

**Оборудование**: компьютерный класс.

**Программное обеспечение**: СУБД MongoDB 4+, 6.0.6 (текущая).

**Практическое задание:**

**Практическое задание 8.1.1:**

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

**Практическое задание 8.1.2:**

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

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

**Практическое задание 8.1.3:**

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

**Практическое задание 8.1.4:**

*Вывести список единорогов в обратном порядке добавления.*

**Практическое задание 8.1.5:**

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

**Практическое задание 8.1.6:**

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

**Практическое задание 8.1.7:**

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

**Практическое задание 8.1.8:**

*Найти всех единорогов, не имеющих ключ vampires.*

**Практическое задание 8.1.9:**

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

**Практическое задание 8.2.1:**

*1) 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"}}
```

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

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

**Практическое задание 8.2.2:**

*1) Сформировать функцию для вывода списка самцов единорогов.*

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

*3) Вывести результат, используя forEach.*

*4) Содержание коллекции единорогов unicorns:*

**Практическое задание 8.2.3:**

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

**Практическое задание 8.2.4:**

*Вывести список предпочтений.*

**Практическое задание 8.2.5:**

*Посчитать количество особей единорогов обоих полов.*

**Практическое задание 8.2.6:**

*1. Выполнить команду:*

> db.unicorns.save({name: 'Barny', loves: ['grape'],

weight: 340, gender: 'm'})

***2.*** *Проверить содержимое коллекции unicorns.*

**Практическое задание 8.2.7:**

*1. Для самки единорога* Ayna *внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.*

*2. Проверить содержимое коллекции unicorns.*

**Практическое задание 8.2.8:**

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

2. *Проверить содержимое коллекции unicorns.*

**Практическое задание 8.2.9:**

*1. Всем самцам единорогов увеличить количество убитых вапмиров на 5.*

*2. Проверить содержимое коллекции unicorns.*

**Практическое задание 8.2.10:**

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

*2. Проверить содержимое коллекции towns.*

**Практическое задание 8.2.11:**

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

*2. Проверить содержимое коллекции unicorns.*

**Практическое задание 8.2.12:**

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

*2. Проверить содержимое коллекции unicorns.*

**Практическое задание 8.2.13:**

*1) Создайте коллекцию towns, включающую следующие документы:*

{name: "Punxsutawney ",
popujatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: ["phil the groundhog"],
mayor: {
  name: "Jim Wehrle"
}}

{name: "New York",
popujatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
  name: "Michael Bloomberg",
party: "I"}}

{name: "Portland",
popujatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
  name: "Sam Adams",
party: "D"}}

*2) Удалите документы с беспартийными мэрами.*
*3) Проверьте содержание коллекции.*
*4) Очистите коллекцию.*
*5) Просмотрите список доступных коллекций.*

**Практическое задание 8.3.1:**

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

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

*3) Проверьте содержание коллекции едиорогов.*

*4) Содержание коллекции единорогов unicorns:*

**Практическое задание 8.3.2:**

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

*Содержание коллекции единорогов unicorns:*

**Практическое задание 8.3.3:**

*1) Получите информацию о всех индексах коллекции unicorns .*

*2) Удалите все индексы, кроме индекса для идентификатора.*

*3) Попытайтесь удалить индекс для идентификатора.*

**Практическое задание 8.3.4:**

*1) Создайте объемную коллекцию numbers, задействовав курсор:*

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

*2) Выберите последних четыре документа.*

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

*4) Создайте индекс для ключа value.*

*5) Получите информацию о всех индексах коллекции numbres.*

*6) Выполните запрос 2.*

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

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

## Выполнение:

## Практическое задание 8.1.1:

*Создайте базу данных learn.*

```
> use learn
< switched to db learn
Atlas atlas-g3ik2e-shard-0 [primary] learn>
```

*Заполните коллекцию единорогов unicorns:*

```
> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c9380f64f9ffd3943f7e8")
    }
  }
> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c938cf64f9ffd3943f7e9")
    }
  }
> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c9394f64f9ffd3943f7ea")
    }
  }
```

```
> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c93a0f64f9ffd3943f7eb")
    }
  }
> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c93b2f64f9ffd3943f7ec")
    }
  }
> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c93bcf64f9ffd3943f7ed")
    }
  }
> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c93c7f64f9ffd3943f7ee")
    }
```

```
> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c93d0f64f9ffd3943f7ef")
    }
  }
> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c93dbf64f9ffd3943f7f0")
    }
  }
> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c93e3f64f9ffd3943f7f1")
    }
  }
> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("649c93ecf64f9ffd3943f7f2")
```

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

```
> doc = {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
< {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
> db.unicorns.insertOne(doc)
< {
    acknowledged: true,
    insertedId: ObjectId("649c94e9f64f9ffd3943f7f3")
  }
```

*Проверьте содержимое коллекции с помощью метода find.*

```
> db.unicorns.find()
< {
    _id: ObjectId("649c9380f64f9ffd3943f7e8"),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId("649c938cf64f9ffd3943f7e9"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

```
{
  _id: ObjectId("649c9394f64f9ffd3943f7ea"),
  name: 'Unicrom',
  loves: [
    'energon',
    'redbull'
  ],
  weight: 984,
  gender: 'm',
  vampires: 182
}
{
  _id: ObjectId("649c93a0f64f9ffd3943f7eb"),
  name: 'Roooooodles',
  loves: [
    'apple'
  ],
  weight: 575,
  gender: 'm',
  vampires: 99
}
```

```
  {
    _id: ObjectId("649c93b2f64f9ffd3943f7ec"),
    name: 'Solnara',
    loves: [
      'apple',
      'carrot',
      'chocolate'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
  {
    _id: ObjectId("649c93bcf64f9ffd3943f7ed"),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
```

```
{
  _id: ObjectId("649c93c7f64f9ffd3943f7ee"),
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
{
  _id: ObjectId("649c93d0f64f9ffd3943f7ef"),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
```

и т д

## Практическое задание 8.1.2:

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

```
> db.unicorns.find({gender: 'f'}, {_id: 0, name: 1}).sort({name: 1}).limit(3).toArray()
< [ { name: 'Aurora' }, { name: 'Ayna' }, { name: 'Leia' } ]
```

```
> db.unicorns.find({gender: 'm'}, {_id: 0, name: 1}).sort({name: 1}).toArray()
< [
    { name: 'Dunx' },
    { name: 'Horny' },
    { name: 'Kenny' },
    { name: 'Pilot' },
    { name: 'Raleigh' },
    { name: 'Roooooodles' },
    { name: 'Unicrom' }
  ]
```

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

```
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
< {
    _id: ObjectId("649c938cf64f9ffd3943f7e9"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
< {
    _id: ObjectId("649c938cf64f9ffd3943f7e9"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

**Практическое задание 8.1.3:**

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

```
> db.unicorns.find({gender: 'f'}, {loves: false, gender: false})
< {
    _id: ObjectId("649c938cf64f9ffd3943f7e9"),
    name: 'Aurora',
    weight: 450,
    vampires: 43
  }
  {
    _id: ObjectId("649c93b2f64f9ffd3943f7ec"),
    name: 'Solnara',
    weight: 550,
    vampires: 80
  }
  {
    _id: ObjectId("649c93bcf64f9ffd3943f7ed"),
    name: 'Ayna',
    weight: 733,
    vampires: 40
  }
  {
    _id: ObjectId("649c93dbf64f9ffd3943f7f0"),
    name: 'Leia',
    weight: 601,
    vampires: 33
  }
  {
    _id: ObjectId("649c93ecf64f9ffd3943f7f2"),
    name: 'Nimue',
```

**Практическое задание 8.1.4:**

*Вывести список единорогов в обратном порядке добавления.*

```
> db.unicorns.find({}, {_id: false, name: true}).sort({ $natural: -1 })
< {
    name: 'Dunx'
  }
  {
    name: 'Nimue'
  }
  {
    name: 'Pilot'
  }
  {
    name: 'Leia'
  }
  {
    name: 'Raleigh'
  }
  {
    name: 'Kenny'
  }
  {
    name: 'Ayna'
  }
  {
    name: 'Solnara'
  }
  {
    name: 'Roooooodles'
  }
```

## Практическое задание 8.1.5:

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

```
> db.unicorns.find({}, {_id: false, loves: { $slice: 1 }})
< {
    name: 'Horny',
    loves: [
      'carrot'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    name: 'Aurora',
    loves: [
      'carrot'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    name: 'Unicrom',
    loves: [
      'energon'
    ],
    weight: 984,
    gender: 'm',
```

```
  {
    name: 'Roooooodles',
    loves: [
      'apple'
    ],
    weight: 575,
    gender: 'm',
    vampires: 99
  }
  {
    name: 'Solnara',
    loves: [
      'apple'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
  {
    name: 'Ayna',
    loves: [
      'strawberry'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
```

## Практическое задание 8.1.6:

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

```
> db.unicorns.find({ gender: "f", weight: { $gte: 500, $lt: 700 } }, {_id: false})
< {
    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'
```

## Практическое задание 8.1.7:

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

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

## Практическое задание 8.1.8:

*Найти всех единорогов, не имеющих ключ vampires.*

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

## Практическое задание 8.1.9:

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

```
> 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'
    ]
  }
```

## Практическое задание 8.2.1:

*Создайте коллекцию towns, включающую следующие документы:*

```
> db.createCollection("towns")
< { ok: 1 }
```

```
    acknowledged: true,
    insertedId: ObjectId("649d113af64f9ffd3943f7f4")
  }
> db.towns.insertOne({name: "New York",
  populatiuon: 22200000,
  last_sensus: ISODate("2009-07-31"),
  famous_for: ["status of liberty", "food"],
  mayor: {
     name: "Michael Bloomberg",
  party: "I"}}
  )
< {
    acknowledged: true,
    insertedId: ObjectId("649d1154f64f9ffd3943f7f5")
  }
> db.towns.insertOne({name: "Portland",
  populatiuon: 528000,
  last_sensus: ISODate("2009-07-20"),
  famous_for: ["beer", "food"],
  mayor: {
     name: "Sam Adams",
  party: "D"}}
  )
< {
    acknowledged: true,
    insertedId: ObjectId("649d116ff64f9ffd3943f7f6")
  }
```

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

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

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

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

## Практическое задание 8.2.2:

***Сформировать функцию для вывода списка самцов единорогов.***

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

*Вывести результат, используя forEach.*

*В compass не работает*

```
> fn = function(){return this.gender == 'm';}
< [Function: fn]
> cursor.forEach(function(obj) {print(obj)});
✖ ▸ ReferenceError: cursor is not defined
> var cursor = db.unicorns.find({'$where': fn }).sort({ name: 1 }).limit(2); null;
)var cursor = db.unicorns.find({'$where': fn }).sort({ name: 1 }).limit(2); null;
```

```
[learn> male = function() {return this.gender == 'm'}
 [Function: male]
```

```
learn> var cursor = db.unicorns.find({ $where: function() { return this.gender === 'm'; } }).limit(2).sort({ name
: 1 });
```

```
learn> cursor.forEach(function(obj) {
...   print(obj.name);
... });
Dunx
Horny
```

## Практическое задание 8.2.3:

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

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

## Практическое задание 8.2.4:

*Вывести список предпочтений.*

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

## Практическое задание 8.2.5:

*Посчитать количество особей единорогов обоих полов.*

```
> db.unicorns.aggregate({ $group: { _id: "$gender", counta: { $sum: 1 } } })
< {
    _id: 'm',
    counta: 7
  }
  {
    _id: 'f',
    counta: 5
  }
```

## Практическое задание 8.2.6:

*Выполнить команду:*

*db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'}) (save устаревшая команда)*

```
> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
< {
    acknowledged: true,
    insertedId: ObjectId("649d184fca44d739ac40bdec")
  }
```

```
> db.unicorns.find({}, { _id: false })
```

```
{
  name: 'Barny',
  loves: [
    'grape'
  ],
  weight: 340,
  gender: 'm'
}
```

## Практическое задание 8.2.7:

*Для самки единорога* Ayna *внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.*

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

```
{
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 800,
  gender: 'f',
  vampires: 51
}
```

## Практическое задание 8.2.8:

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

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

**Практическое задание 8.2.9:**

*Всем самцам единорогов увеличить количество убитых вапмиров на 5.*

*Проверить содержимое коллекции unicorns.*

```
> db.unicorns.find({ gender: "m" }, { _id: false })
< {
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
```

```
> db.unicorns.updateMany({ gender: "m" }, { $inc: { vampires: 5 } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 8,
    modifiedCount: 8,
    upsertedCount: 0
  }
> db.unicorns.find({ gender: "m" }, { _id: false })
< {
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 68
  }
```

**Практическое задание 8.2.10:**

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

*Проверить содержимое коллекции towns.*

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

```
{
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams'
  }
}
```

## Практическое задание 8.2.11:

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

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

## Практическое задание 8.2.12:

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

*Проверить содержимое коллекции unicorns.*

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

## Практическое задание 8.2.13:

*Создайте коллекцию towns, включающую следующие документы:*

```
{
  acknowledged: true,
  insertedId: ObjectId("649d1c98ca44d739ac40bded")
}
db.towns.insertOne({name: "New York",
popujatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
   name: "Michael Bloomberg",
party: "I"}}
)
{
  acknowledged: true,
  insertedId: ObjectId("649d1cbcca44d739ac40bdee")
}
db.towns.insertOne({name: "Portland",
popujatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
   name: "Sam Adams",
party: "D"}}
)
{
  acknowledged: true,
  insertedId: ObjectId("649d1cceca44d739ac40bdef")
}
```

*Удалите документы с беспартийными мэрами.*

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

*Проверьте содержание коллекции. Кого удалили:*

```
{
  name: 'Punxsutawney ',
  populatiuon: 6200,
  last_sensus: 2008-01-31T00:00:00.000Z,
  famous_for: [
    ''
  ],
  mayor: {
    name: 'Jim Wehrle'
  }
}
```

```
{
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams'
  }
}
```

```
{
  name: 'Punxsutawney ',
  popujatiuon: 6200,
  last_sensus: 2008-01-31T00:00:00.000Z,
  famous_for: [
    'phil the groundhog'
  ],
  mayor: {
    name: 'Jim Wehrle'
  }
}
```

*Очистите коллекцию.*

```
> db.towns.drop()
< true
```

*Просмотрите список доступных коллекций.*

```
> db.getCollectionNames()
< [ 'unicorns' ]
```

**Практическое задание 8.3.1:**

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

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

*Проверьте содержание коллекции едиорогов.*

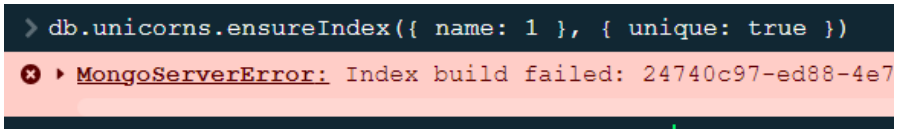
```
> db.areas.find()
< {
    _id: '1',
    name: 'area_1'
  }
  {
    _id: '2',
    name: 'area_2'
  }
```

```
> db.unicorns.updateOne({ name: "Dunx" }, { $set: { zone: { $ref: "zone", $id: "1" }}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({}, { _id: false })
```

```
{
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165,
  zone: DBRef("zone", '1')
}
```

**Практическое задание 8.3.2:**

*Проверьте, можно ли задать для коллекции  unicorns индекс для ключа name с флагом* unique. *(нельзя)*

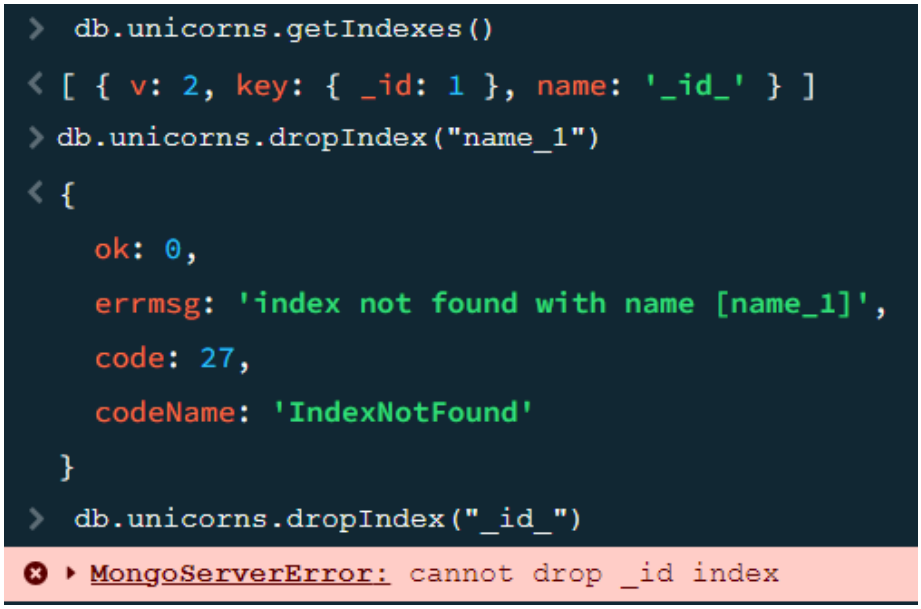



**Практическое задание 8.3.3:**

*Получите информацию о всех индексах коллекции unicorns.*

*Удалите все индексы, кроме индекса для идентификатора.*

*Попытайтесь удалить индекс для идентификатора.*



**Практическое задание 8.3.4:**

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

*(долго думает)*

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

{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477b9e20e5aba34a3f77062") }
}
```

*Выберите последних четыре документа.*

```
learn> db.numbers.find({}).skip(99996)
[
  { _id: ObjectId("6477b9e20e5aba34a3f7705f"), value: 99996 },
  { _id: ObjectId("6477b9e20e5aba34a3f77060"), value: 99997 },
  { _id: ObjectId("6477b9e20e5aba34a3f77061"), value: 99998 },
  { _id: ObjectId("6477b9e20e5aba34a3f77062"), value: 99999 }
]
```

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

```
learn> db.numbers.explain('executionStats').find({}).skip(99996)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'SKIP',
      skipAmount: 0,
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 43,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      stage: 'SKIP',
      nReturned: 4,
      executionTimeMillisEstimate: 1,
      works: 100002,
      advanced: 4,
      needTime: 99997,
      needYield: 0,
      saveState: 100,
      restoreState: 100,
      isEOF: 1,
      skipAmount: 0,
```


На выполнение запроса понадобилось 43 миллисекунды.

*Создайте индекс для ключа value.*

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

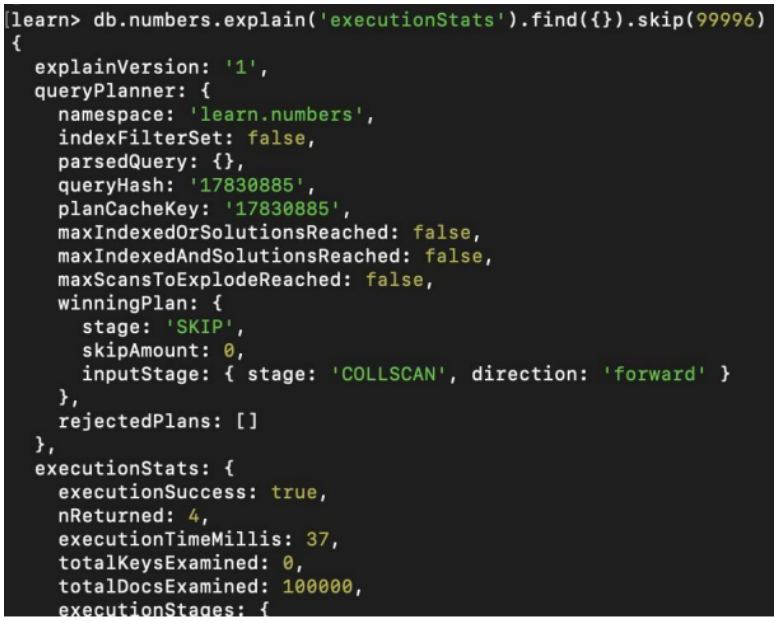
*Получите информацию о всех индексах коллекции numbres.*

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

*Выполните запрос 2.*

```
learn> db.numbers.find({}).skip(99996)
[
  { _id: ObjectId("6477b9e20e5aba34a3f7705f"), value: 99996 },
  { _id: ObjectId("6477b9e20e5aba34a3f77060"), value: 99997 },
  { _id: ObjectId("6477b9e20e5aba34a3f77061"), value: 99998 },
  { _id: ObjectId("6477b9e20e5aba34a3f77062"), value: 99999 }
]
```

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



```
learn> db.numbers.explain('executionStats').find({}).skip(99996)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'SKIP',
      skipAmount: 0,
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 37,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
```

На выполнение запроса с индексом, установленным на value, понадобилось

37 миллисекунд.

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

При небольшом количестве данных в коллекции разница во времени почти не ощутима, однако, если увеличить количество данных, то запросы с индексом будут выполняться быстрее.