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»**

Автор: Шалунов Андрей Ильич

Факультет: ИКТ

Группа: К3240 Преподаватель:

Говорова М.М.

Санкт-Петербург 2023

Оглавление

Цель работы

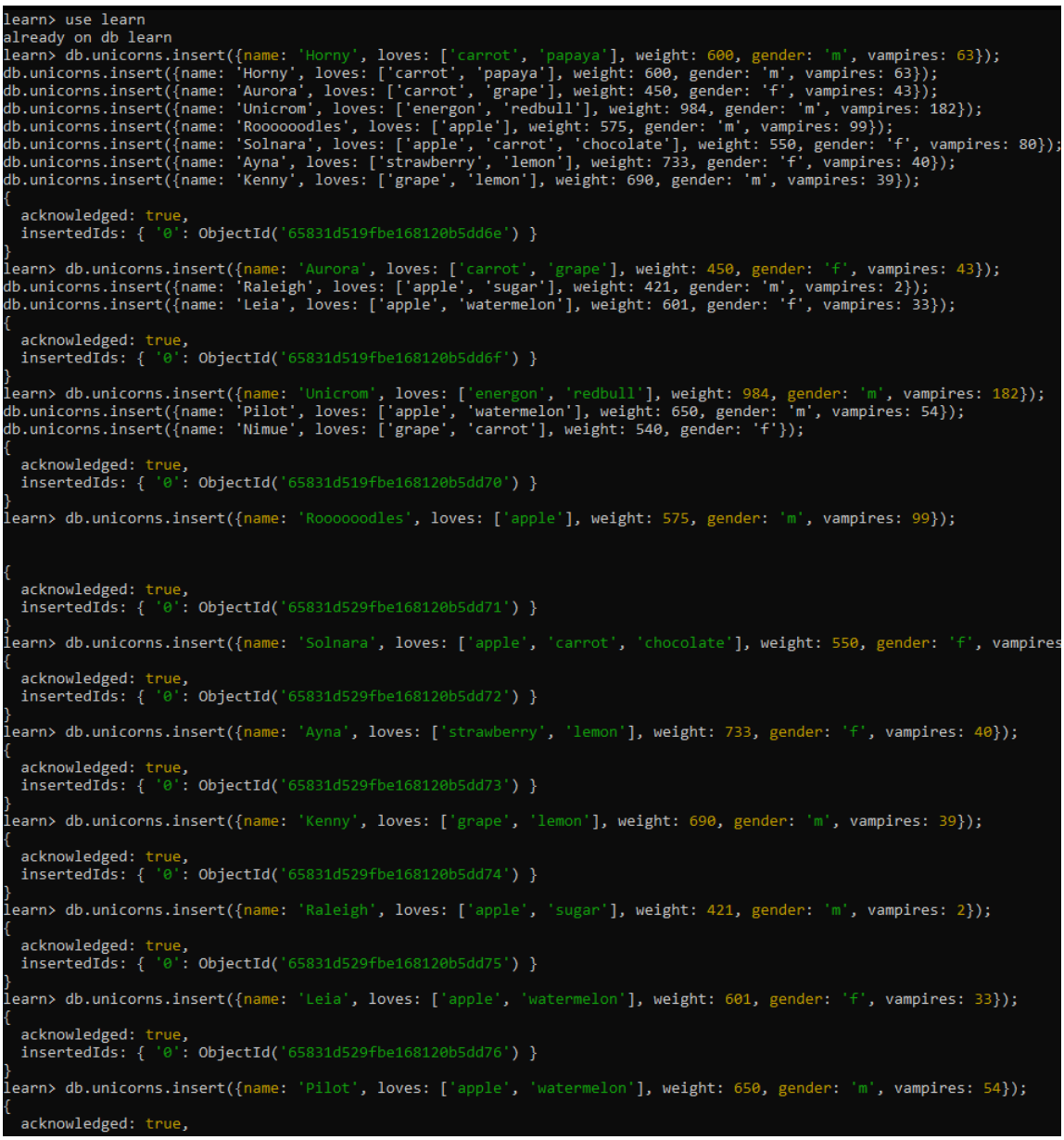
Овладеть практическими навыками работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.

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

2.1 ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ

Практическое задание 2.1.1:

```
use learn
db.unicorns.insert({name: 'Horny', loves: ['carrot', 'papaya'], weight: 600, gender: 'm', vampires: 63});
db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
db.unicorns.insert({name: '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'});
```



```
learn> use learn
already on db learn
learn> db.unicorns.insert({name: 'Horny', loves: ['carrot', 'papaya'], weight: 600, gender: 'm', vampires: 63});
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});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65831d519fbe168120b5dd6e') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
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});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65831d519fbe168120b5dd6f') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
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'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65831d519fbe168120b5dd70') }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});


{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65831d529fbe168120b5dd71') }
}
learn> db.unicorns.insert({name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65831d529fbe168120b5dd72') }
}
learn> db.unicorns.insert({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65831d529fbe168120b5dd73') }
}
learn> db.unicorns.insert({name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65831d529fbe168120b5dd74') }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65831d529fbe168120b5dd75') }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65831d529fbe168120b5dd76') }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
```

```
const document = {
  name: 'Dunx',
  loves: ['grape', 'watermelon'],
  weight: 704,
  gender: 'm',
  vampires: 165
};

db.unicorns.insert(document);

db.unicorns.find();
```

Практическое задание 2.2.1

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

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

```
learn> db.unicorns.find({ gender: 'm' }).sort({ name: 1 });
[
  {
    _id: ObjectId('65831d819fbe168120b5dd80'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65831d519fbe168120b5dd6e'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd74'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd77'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7f'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd75'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd71'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
```

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

```
learn> db.unicorns.find({ gender: 'm' }).sort({ name: 1 });
[
  {
    _id: ObjectId('65831d819fbe168120b5dd80'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65831d519fbe168120b5dd6e'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd74'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd77'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7f'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd75'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
```

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

```
learn> db.unicorns.find({ gender: 'f', loves: 'carrot' }).limit(1);
[
  {
    _id: ObjectId('65831d519fbe168120b5dd6f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

```
learn> db.unicorns.findOne({ gender: 'f', loves: 'carrot' });
{
  _id: ObjectId('65831d519fbe168120b5dd6f'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```

Практическое задание 2.2.2

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

db.unicorns.find({ gender: 'm' }, { loves: 0, weight: 0 }).sort({ $natural: 1 });

```
learn> db.unicorns.find({ gender: 'm' }, { loves: 0, weight: 0 }).sort({ $natural: 1 }
[
  {
    _id: ObjectId('65831d519fbe168120b5dd6e'),
    name: 'Horny',
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65831d519fbe168120b5dd70'),
    name: 'Unicrom',
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd71'),
    name: 'Roooooodles',
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd74'),
    name: 'Kenny',
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd75'),
    name: 'Raleigh',
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd77'),
    name: 'Pilot',
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7a'),
    name: 'Unicrom',
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7b'),
    name: 'Roooooodles',
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7e'),
    name: 'Raleigh',
    gender: 'm',
    vampires: 2
  },
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

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

```
learn> db.unicorns.find().sort({ $natural: -1 });
[
  {
    _id: ObjectId('65831d819fbe168120b5dd80'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7f'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7d'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7c'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7b'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7a'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
```

Практическое задание 2.1.4

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

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

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

Практическое задание 2.3.1:

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

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

```
learn> db.unicorns.find({ gender: 'f', weight: { $gte: 500, $lte: 700 } }, { _id: 0, name: 1, gender: 1, weight: 1 });
[
  { name: 'Solnara', weight: 550, gender: 'f' },
  { name: 'Leia', weight: 601, gender: 'f' },
  { name: 'Nimue', weight: 540, gender: 'f' },
  { name: 'Solnara', weight: 550, gender: 'f' }
]
```

Практическое задание 2.3.2:

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: 0, name: 1, gender: 1, weight: 1, loves: 1 }

);

```
learn> db.unicorns.find(
...   { gender: 'm', weight: { $gte: 500 } , loves: { $all: ['grape', 'lemon'] } },
...   { _id: 0, name: 1, gender: 1, weight: 1, loves: 1 }
... );
[
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm'
  }
]
learn>
```

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

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

```
learn> db.unicorns.find({ vampires: { $exists: false } });
[
  {
    _id: ObjectId('65831d529fbe168120b5dd78'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.4:

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

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

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

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:

Создайте коллекцию towns, включающую следующие документы:

```
db.towns.insertMany([
  {
    name: "Punxsutawney",
    population: 6200,
    last_sensus: ISODate("2008-01-31"),
    famous_for: [""],
    mayor: {
      name: "Jim Wehrle"
    }
  },
  {
    name: "New York",
    population: 22200000,
    last_sensus: ISODate("2009-07-31"),
    famous_for: ["Statue 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"
    }
  }
]);
```

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: 0, name: 1, "mayor.name": 1, "mayor.party": 1 });

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

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: 0, name: 1, "mayor.name": 1 });

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

Практическое задание 3.1.2:

Сформировать функцию для вывода списка самцов единорогов.

fn_uni = function() {return db.unicorns.find({gender: "m"}) }

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

var cursor = fn_uni(); null;

cursor.limit(2).sort({name: 1}); null;

```
learn> var cursor = fn_uni(); null;
null
learn> cursor.limit(2).sort({name: 1}); null;
null
learn>
```

Вывести результат, используя forEach.

cursor.forEach(function(obj) {print(obj)})

```
learn> cursor.forEach(function(obj) {print(obj)})
{
  _id: ObjectId('65831d819fbe168120b5dd80'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('65831d519fbe168120b5dd6e'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}

learn>
```

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

```
}

learn> db.unicorns.find({ gender: 'f', weight: { $gte: 500, $lte: 600 } }).count()
3
learn>
```

Практическое задание 3.2.2:

Вывести список предпочтений.

db.unicorns.distinct('loves')

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

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

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

```
learn> db.unicorns.aggregate([{$group:{_id:"$gender",count:{$sum:1}}}])
[ { _id: 'm', count: 11 }, { _id: 'f', count: 8 } ]
learn>
```

Практическое задание 3.3.2:

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

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

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

Проверить содержимое коллекции `unicorns`.

```
learn> db.unicorns.find({ name: 'Ayna' });
[
  {
    _id: ObjectId('65831d529fbe168120b5dd73'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7d'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
]
learn>
```

Практическое задание 3.3.3:

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

Проверить содержимое коллекции `unicorns`.

db.unicorns.updateOne({ name: 'Raleigh', gender: 'm' }, { $set: { loves: ['redbull'] } });

```
learn> db.unicorns.updateOne({name: 'Raleigh', gender: 'm'}, {$set:{love:['redbull']}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
learn>
```

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

```
learn> db.unicorns.find({ name: 'Raleigh' });
[
  {
    _id: ObjectId('65831d529fbe168120b5dd75'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2,
    love: [ 'redbull' ]
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
learn>
```

Практическое задание 3.3.4:

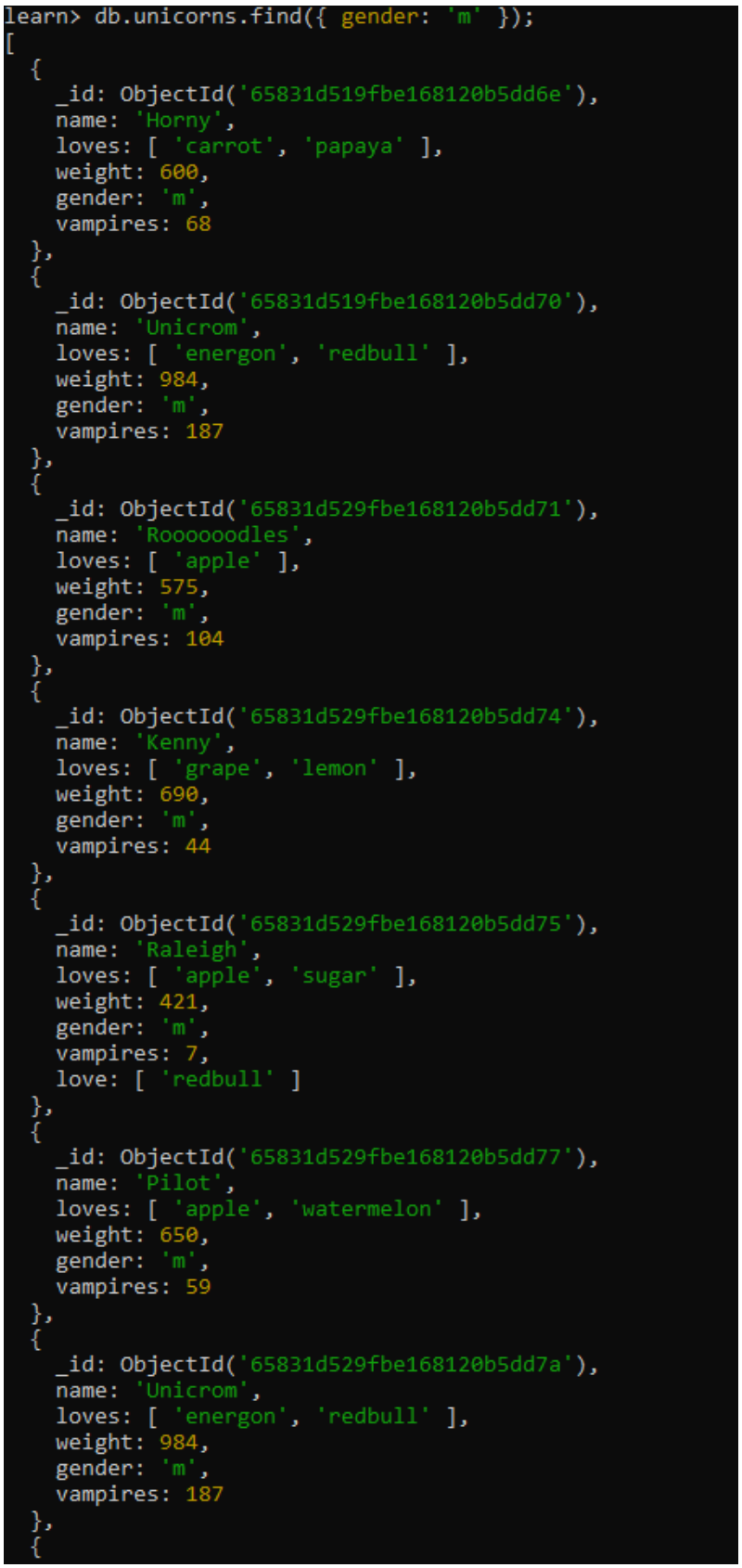
Всем самцам единорогов увеличить количество убитых вапмиров на 5.

Проверить содержимое коллекции `unicorns`.

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

```
learn> db.unicorns.updateMany({ gender: 'm' }, { $inc: { vampires: 5 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 11,
  modifiedCount: 11,
  upsertedCount: 0
}
learn>
```

db.unicorns.find({ gender: 'm' });



Практическое задание 3.3.5:

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

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

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

```
learn>
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Проверить содержимое коллекции `towns`.

```
learn> db.towns.find({ name: 'Portland' });
[
  {
    _id: ObjectId('65832ac39fbe168120b5dd83'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
learn>
```

Практическое задание 3.3.6:

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

Проверить содержимое коллекции `unicorns`.

db.unicorns.updateOne({ name: 'Pilot', gender: 'm' }, { $push: { loves: 'chocolate' } });

```
learn> db.unicorns.updateOne({ name: 'Pilot', gender: 'm' }, { $push: { loves: 'chocolate' } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

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

```
learn> db.unicorns.find({ name: 'Pilot' });
[
  {
    _id: ObjectId('65831d529fbe168120b5dd77'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd7f'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
learn>
```

Практическое задание 3.3.7:

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

Проверить содержимое коллекции `unicorns`.

db.unicorns.updateOne({ name: 'Aurora', gender: 'f' }, { $addToSet: { loves: { $each: ['sugar', 'lemons'] } } });

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

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

```
learn> db.unicorns.find({ name: 'Aurora' });
[
  {
    _id: ObjectId('65831d519fbe168120b5dd6f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
```

Практическое задание 3.4.1:

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"}}
```

```
learn> db.unicorns.find({ name: 'Aurora' });
[
  {
    _id: ObjectId('65831d519fbe168120b5dd6f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65831d529fbe168120b5dd79'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn> db.towns.insert([
...   {
...     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('658349e39fbe168120b5dd84'),
    '1': ObjectId('658349e39fbe168120b5dd85'),
    '2': ObjectId('658349e39fbe168120b5dd86')
  }
}
learn>
```

Удалите документы с беспартийными мэрами.
db.towns.remove({ "mayor.party": { $exists: false } });

Проверьте содержание коллекции.

```
learn> db.towns.find();
[
  {
    _id: ObjectId('65832ac39fbe168120b5dd82'),
    name: 'New York',
    population: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'Statue of Liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('658349e39fbe168120b5dd85'),
    name: 'New York',
    population: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('658349e39fbe168120b5dd86'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

Очистите коллекцию.

db.towns.remove({});

Просмотрите список доступных коллекций.

```
learn> db.towns.remove({});
{ acknowledged: true, deletedCount: 3 }
learn> show collections;
towns
unicorns
learn>
```

4 ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

4.1 ССЫЛКИ В БД

Практическое задание 4.1.1:

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

db.habitat.insert({ _id: 'forest', full_name: 'Enchanted Forest', description: 'A magical forest where unicorns thrive.' });

db.habitat.insert({ _id: 'mountain', full_name: 'Mystic Mountain', description: 'A mystical mountain range, home to many unicorns.' });

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

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

db.unicorns.update({name: 'Aurora '}, {$set: {habitat: {$ref:"habitats", $id: "mountain"}}});

```
learn> db.unicorns.update({name: 'Horny'}, {$set: {habitat: {$ref:"habitats", $id: "forest"}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.update({name: 'Aurora        '}, {$set: {habitat: {$ref:"habitats", $id: "mountain"}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
learn>
```

Проверьте содержание коллекции едиорогов.

```
}
learn> db.unicorns.find();
[
  {
    _id: ObjectId('6583572f9fbe168120b5dd87'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63,
    habitat: DBRef('habitats', 'forest')
  },
  {
    _id: ObjectId('6583572f9fbe168120b5dd88'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef('habitats', 'mountain')
  },
  {
```

Содержание 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', 44), 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'});

db.unicorns.insert {name: 'Dunx', loves: ['grape', 'watermelon'], weight:
704, gender: 'm', vampires: 165}
```

Практическое задание 4.2.1:

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

-Создать можно

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

При попытке добавить единорога с существующим именем возникнет ошибка

```
learn> db.unicorns.insert({name: 'Horny', dob: new Date(1992,2,13,7,47), loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
Uncaught:
MongoBulkWriteError: E11000 duplicate key error collection: learn.unicorns index: name_1 dup key: { name: "Horny" }
Result: BulkWriteResult {
  insertedCount: 0,
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {},
  insertedIds: {}
}
Write Errors: [
  WriteError {
    err: {
      index: 0,
      code: 11000,
      errmsg: 'E11000 duplicate key error collection: learn.unicorns index: name_1 dup key: { name: "Horny" }',
      errInfo: undefined,
      op: {
        name: 'Horny',
        dob: ISODate('1992-03-13T04:47:00.000Z'),
        loves: [ 'carrot', 'papaya' ],
        weight: 600,
        gender: 'm',
        vampires: 63,
        _id: ObjectId('65835c659fbe168120b5dd9f')
      }
    }
  }
]
learn>
```

Практическое задание 4.3.1:

Получите информацию о всех индексах коллекции `unicorns` .

db.unicorns.getIndexes();

```
MongoServerError: ns does not exist: learn.numbers
learn> db.unicorns.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
learn>
```

Удалите все индексы, кроме индекса для идентификатора.

```
db.unicorns.getIndexes().forEach(function(index) {
   if (index.name !== "_id_") {
      db.unicorns.dropIndex(index.name);
   }
});
```

Попытайтесь удалить индекс для идентификатора.

db.unicorns.dropIndex("_id_");

```
learn> db.unicorns.dropIndex("_id_");
MongoServerError: cannot drop _id index
learn>
```

Практическое задание 4.4.1:

Создайте объемную коллекцию `numbers`*, задействовав курсор:*

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

```
test> for (let i = 0; i < 100000; i++) {
...     db.numbers.insertOne({ value: i });
... }
```

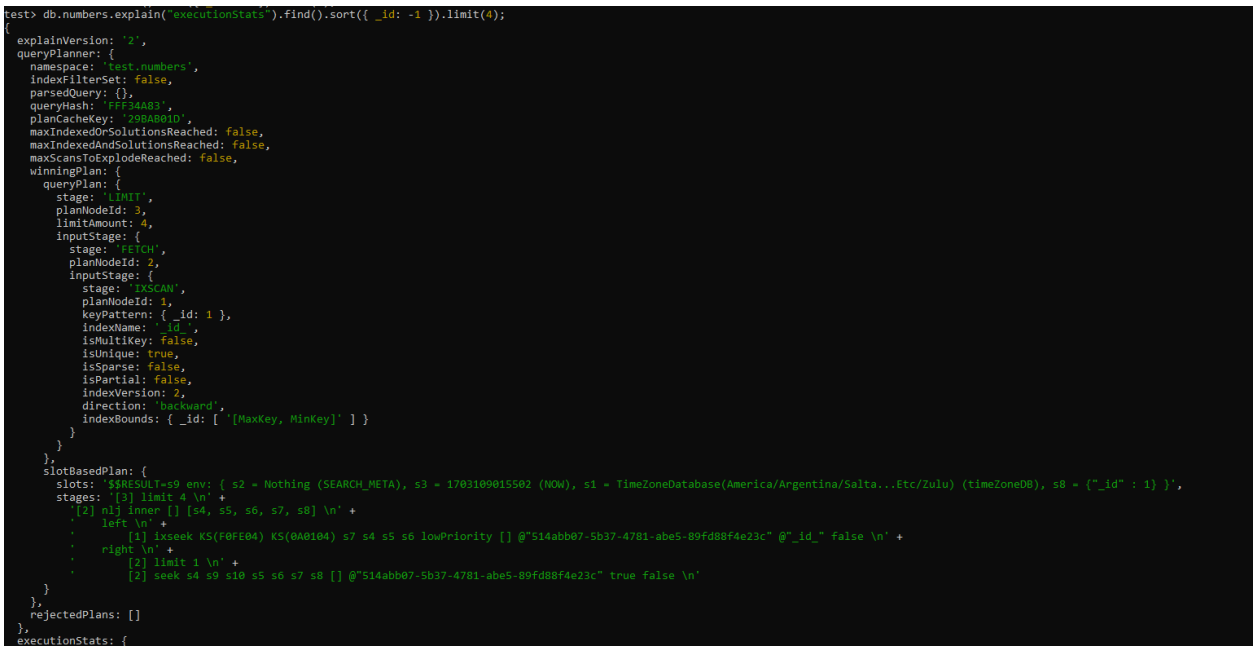
```
{
  acknowledged: true,
  insertedId: ObjectId('658360607e6a5beedad8a4f7')
}
test>
```

Выберите последних четыре документа.

db.numbers.find().sort({ _id: -1 }).limit(4);

```
test> db.numbers.find().sort({ _id: -1 }).limit(4);
[
  { _id: ObjectId('658360607e6a5beedad8a4f7'), value: 99999 },
  { _id: ObjectId('658360607e6a5beedad8a4f6'), value: 99998 },
  { _id: ObjectId('658360607e6a5beedad8a4f5'), value: 99997 },
  { _id: ObjectId('658360607e6a5beedad8a4f4'), value: 99996 }
]
test>
```

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



executionTimeMillis: 2

Создайте индекс для ключа `value`*.*

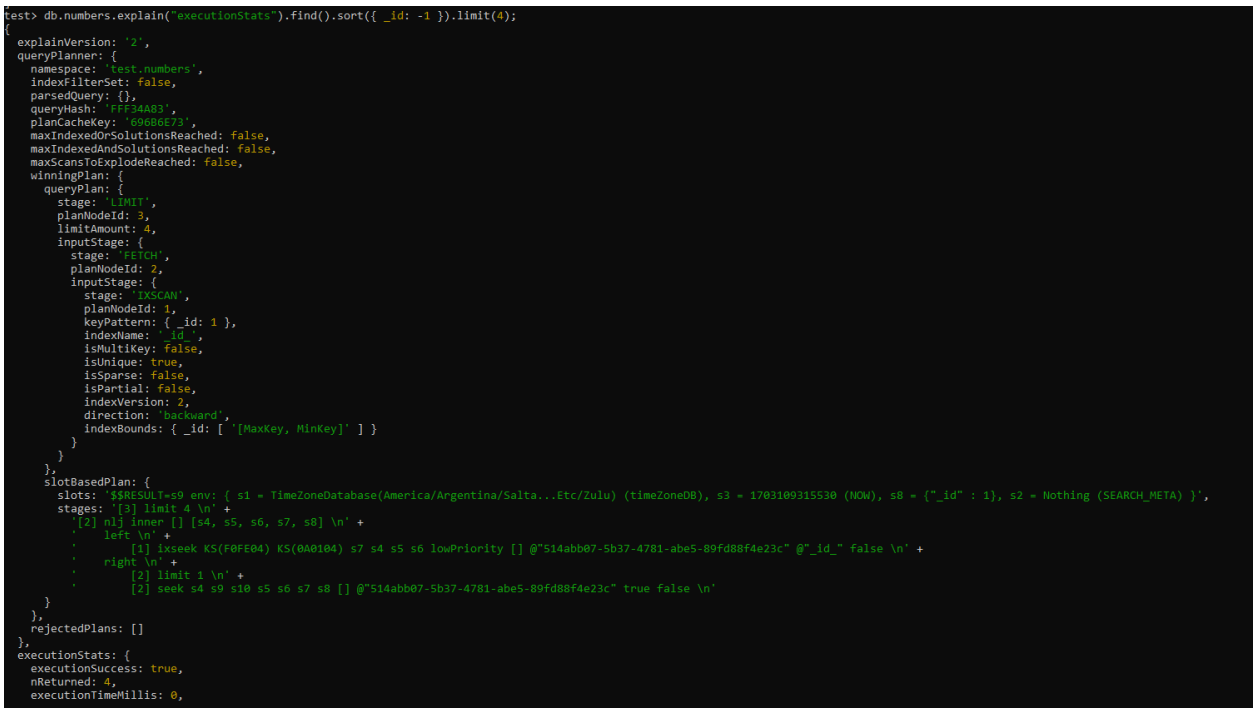
db.numbers.createIndex({ value: 1 });

Получите информацию о всех индексах коллекции `numbres`*.*

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

Выполните запрос 2.

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



executionTimeMillis: 0

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

Запрос с индексами более эффективный, так как значение executionTimeMillis у него меньше.

Вывод

В рамках выполнения лабораторной работы по MongoDB были успешно реализованы ключевые задачи. В ходе работы были проведены разносторонние операции с данными, включая добавление, обновление и удаление информации. Также были проведены эксперименты с индексами для оптимизации запросов, включая создание уникальных индексов для обеспечения уникальности значений.