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

Отчет

по лабораторной работе №6 «**Реализация БД с использованием СУБД MongoDB. Запросы к базе данных»**

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

Автор: Филиппов А.Э.

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

Цель работы:.....3
Практическое задание.....3
Выполнение.....3
Вывод.....10

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.

Оборудование: компьютерный класс.

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

Практическое задание:

Вариант 2 (max - 8 баллов)

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
2. Создать авторский триггер по варианту индивидуального задания.

Выполнение

Практическое задание 2.1.1:

1) Создайте базу данных learn.

```
test> use learn;
switched to db learn
```

2) Заполните коллекцию единорогов unicorns:

```
learn> 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('6570d3f7aa2712043f968313') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f7aa2712043f968314') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f7aa2712043f968315') }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f7aa2712043f968316') }
}
learn> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f7aa2712043f968317') }
}
learn> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f7aa2712043f968318') }
}
learn> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f7aa2712043f968319') }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f7aa2712043f96831a') }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f8aa2712043f96831b') }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f8aa2712043f96831c') }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6570d3f8aa2712043f96831d') }
}
```

3) Используя второй способ, вставьте в коллекцию единорогов документ:

```
learn> var doc = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight:
... 704, gender: 'm', vampires: 165})

learn> db.unicorns.insertOne(doc);
{
  acknowledged: true,
  insertedId: ObjectId('6570d54aaa2712043f96831e')
}
```

4) Проверьте содержимое коллекции с помощью метода find.

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('6570d3f7aa2712043f968313'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968314'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968315'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968316'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968317'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968318'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968319'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6570d3f7aa2712043f96831a'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('6570d3f8aa2712043f96831b'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('6570d3f8aa2712043f96831c'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6570d3f8aa2712043f96831d'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6570d54aaa2712043f96831e'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
```

Практическое задание 2.2.1:

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

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('6570d54aaa2712043f96831e'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968313'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968319'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6570d3f8aa2712043f96831c'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6570d3f7aa2712043f96831a'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968316'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968315'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('6570d3f7aa2712043f968314'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968318'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('6570d3f8aa2712043f96831b'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

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('6570d3f7aa2712043f968314'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId('6570d3f7aa2712043f968314'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Практическое задание 2.2.2:

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: false, gender: false})
[
  {
    _id: ObjectId('6570d3f7aa2712043f968313'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968315'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968316'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968319'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('6570d3f7aa2712043f96831a'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('6570d3f8aa2712043f96831c'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('6570d54aaa2712043f96831e'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
]
```

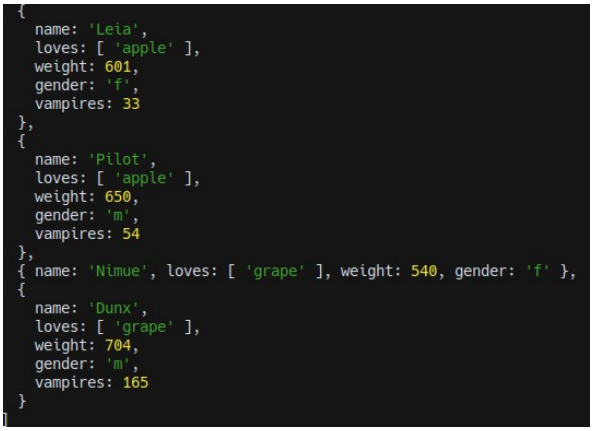
Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('6570d54aaa2712043f96831e'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6570d3f8aa2712043f96831d'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6570d3f8aa2712043f96831c'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6570d3f8aa2712043f96831b'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('6570d3f7aa2712043f96831a'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968319'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968318'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
```

Практическое задание 2.1.4:

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



Практическое задание 2.3.1:

Вывести список 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.

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

Практическое задание 2.3.4:

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

Практическое задание 3.1.1:

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"}}

```
learn> 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"
...         }
...     }
... ]);

{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6570e8d9aa2712043f96831f'),
    '1': ObjectId('6570e8d9aa2712043f968320'),
    '2': ObjectId('6570e8d9aa2712043f968321')
  }
}
learn>
```

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

```
learn> db.towns.find({'mayor.party': "I"}, {name: true, mayor: true});
[
  {
    _id: ObjectId('6570e8d9aa2712043f968320'),
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
```

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

```
learn> db.towns.find({'mayor.party': {$exists: false}}, {name: true, mayor: true});
[
  {
    _id: ObjectId('6570e8d9aa2712043f96831f'),
    name: 'Punxsutawney',
    mayor: { name: 'Jim Wehrle' }
  }
]
```

Практическое задание 3.1.2:

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.

```
learn> function printUnicorns() {
... var cursor = db.unicorns.find({gender: "m"}); null;
... cursor.sort({name: 1}).limit(2); null;
... cursor.forEach((unicorn) => {
... print(unicorn.name);
... });
... }
[Function: printUnicorns]
learn> printUnicorns();
Dunx
Horny
```

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 3.2.2:

Вывести список предпочтений.

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

Практическое задание 3.2.3:

Подсчитать количество особей единорогов обоих полов.

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

Практическое задание 3.3.2:

1. Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.
2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update({name: "Ayna", gender: "f"}, {$set: {weight: 800, vampires: 51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
learn> db.unicorns.find({name: "Ayna"})
[
  {
    _id: ObjectId('6570d3f7aa2712043f968318'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

Практическое задание 3.3.3:

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

2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update({name: "Raleigh", gender: "m"}, {$set: {loves: ["redbull"]}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: "Raleigh"})
[
  {
    _id: ObjectId('6570d3f7aa2712043f96831a'),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Практическое задание 3.3.4:

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

2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.updateMany({gender: "m"}, {$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('6570d3f7aa2712043f968313'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968314'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968315'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968316'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968317'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968318'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  },
```

Практическое задание 3.3.5:

1. Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.
2. Проверить содержимое коллекции towns.

```
learn> db.towns.update({name: "Portland"}, {$set: {'mayo.party': "I"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find()
[
  {
    _id: ObjectId('6570e8d9aa2712043f96831f'),
    name: 'Punxsutawney',
    population: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('6570e8d9aa2712043f968320'),
    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('6570e8d9aa2712043f968321'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: 'I',
    mayo: { partyr: 'I', party: 'I' }
  }
]
```

Практическое задание 3.3.6:

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

Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update({name: "Pilot"}, {$addToSet: {loves: "chocolate"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unciorns.find({name: "Pilot"})

learn> db.unicorns.find({name: "Pilot"})
[
  {
    _id: ObjectId('6570d3f8aa2712043f96831c'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

Практическое задание 3.3.7:

1. Изменить информацию о самке единорога Aurora: теперь она любит еще и сахар, и лимоны.
2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update({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"})
[
  {
    _id: ObjectId('6570d3f7aa2712043f968314'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 3.4.1:

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.

```
learn> db.towns.deleteMany({"mayor.party": {$exists: false}})
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId('6570fb81e87f904e9402b392'),
    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('6570fb81e87f904e9402b393'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

4. Очистите коллекцию.

5. Просмотрите список доступных коллекций.

```
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 2 }
learn> db.towns.find()

learn>
```

Практическое задание 4.1.1:

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

```
learn> db.zones.insert([
...     {
...         _id: "de",
...         name: "Deutschland",
...         desc: "Deine Liebe ist Fluch und Segen"
...     },
...     {
...         _id: "ru",
...         name: "Rossiya",
...         desc: "Moscow never sleeps"
...     },
...     {
...         _id: "kz",
...         name: "Kasachstan",
...         desc: "Fuehlen Sie den Geschmaek der Freiheit"
...     }
... ])
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': 'de', '1': 'ru', '2': 'kz' }
}
learn> db.zones.find()
[
  {
    _id: 'de',
    name: 'Deutschland',
    desc: 'Deine Liebe ist Fluch und Segen'
  },
  { _id: 'ru', name: 'Rossiya', desc: 'Moscow never sleeps' },
  {
    _id: 'kz',
    name: 'Kasachstan',
    desc: 'Fuehlen Sie den Geschmaek der Freiheit'
  }
]
```

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:

```
learn> db.unicorns.update({name: "Horny"}, { $set: { zone: { $ref: "zones", $id: deZone }}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
learn> db.unicorns.update({name: "Aurora"}, { $set: { zone: { $ref: "zones", $id: ruZone }}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
learn> db.unicorns.update({name: "Unicrom"}, { $set: { zone: { $ref: "zones", $id: kzZone }}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('6570d3f7aa2712043f968313'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    zone: DBRef('zones', 'de')
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968314'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    zone: DBRef('zones', 'ru')
  },
  {
    _id: ObjectId('6570d3f7aa2712043f968315'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187,
    zone: DBRef('zones', 'kz')
  }
```

Практическое задание 4.2.1:

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

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

2. Содержание коллекции единорогов unicorns:

db.unicorns.insert({name: 'Horny', dob: new Date(1992,2,13,7,47), loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});

```
db.unicorns.insert({name: 'Aurora', dob: new
Date(1991, 0, 24, 13, 0), loves: ['carrot', 'grape'],
weight: 450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', dob: new
Date(1973, 1, 9, 22, 10), loves: ['energon',
'redbull'], weight: 984, gender: 'm', vampires:
182});

db.unicorns.insert({name: 'Roooooodles', dob:
new Date(1979, 7, 18, 18, 44), loves: ['apple'],
weight: 575, gender: 'm', vampires: 99});

db.unicorns.insert({name: 'Solnara', dob: new
Date(1985, 6, 4, 2, 1), loves:['apple', 'carrot',
'chocolate'], weight:550, gender:'f',
vampires:80});

db.unicorns.insert({name:'Ayna', dob: new
Date(1998, 2, 7, 8, 30), loves: ['strawberry',
'lemon'], weight: 733, gender: 'f', vampires:
40});

db.unicorns.insert({name:'Kenny', dob: new
Date(1997, 6, 1, 10, 42), loves: ['grape', 'lemon'],
weight: 690,  gender: 'm', vampires: 39});

db.unicorns.insert({name: 'Raleigh', dob: new
Date(2005, 4, 3, 0, 57), loves: ['apple', 'sugar'],
weight: 421, gender: 'm', vampires: 2});

db.unicorns.insert({name: 'Leia', dob: new
Date(2001, 9, 8, 14, 53), loves: ['apple',
'watermelon'], weight: 601, gender: 'f',
vampires: 33});

db.unicorns.insert({name: 'Pilot', dob: new
Date(1997, 2, 1, 5, 3), loves: ['apple',
'watermelon'], weight: 650, gender: 'm',
vampires: 54});

db.unicorns.insert ({name: 'Nimue', dob: new
Date(1999, 11, 20, 16, 15), loves: ['grape',
'carrot'], weight: 540, gender: 'f'});

db.unicorns.insert {name: 'Dunx', dob: new
Date(1976, 6, 18, 18, 18), loves: ['grape',
'watermelon'], weight: 704, gender: 'm',
vampires: 165
```

Практическое задание 4.3.1:

1. Получите информацию о всех индексах коллекции unicorns .
2. Удалите все индексы, кроме индекса для идентификатора.
3. Попытайтесь удалить индекс для идентификатора.

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

```
learn> db.unicorns.dropIndexes()
{
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
```

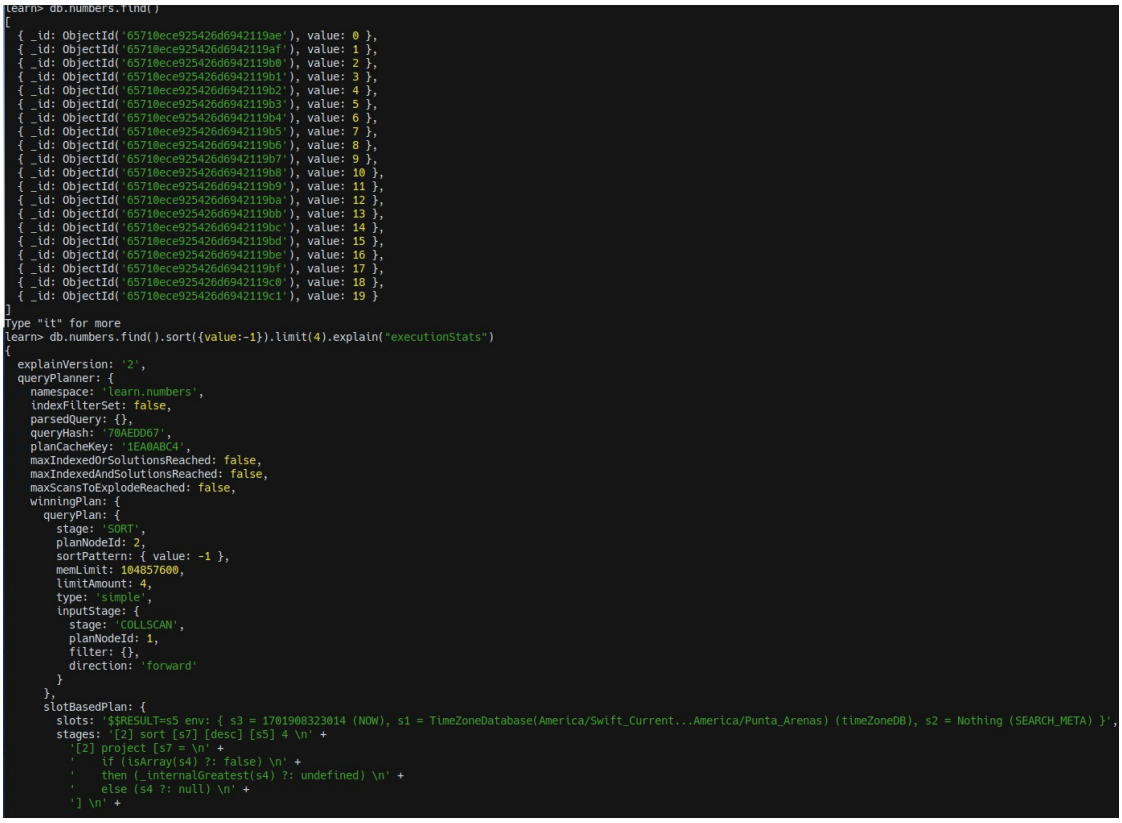
Практическое задание 4.4.1:

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 = 61)



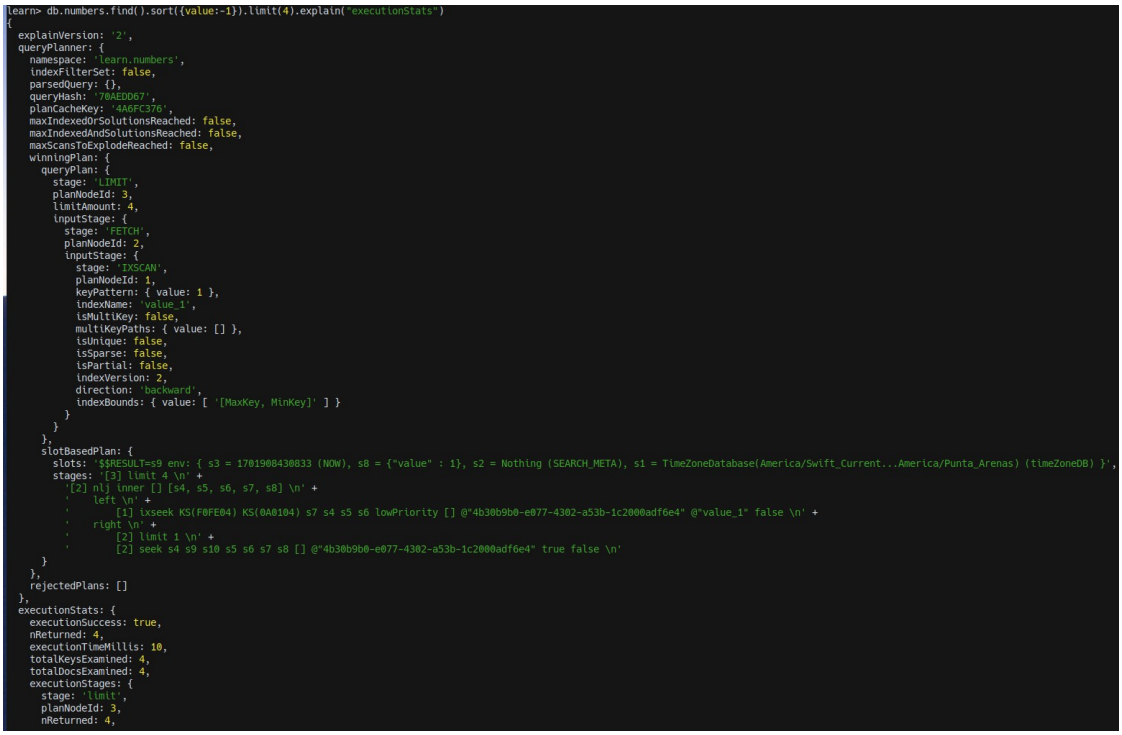
4. Создайте индекс для ключа value.

5. Получите информацию о всех индексах коллекции numbres.

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

6.

7. Выполните запрос 2.



8. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса? ExecutionTimeMillis = 10
9. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен? Ответ: эффективен запрос с использованием индексов

Вывод

В ходе лабораторной работы была освоена работа в MongoDB с кликерами, агрегациями, ссылками и индексами. Были закреплены необходимые теоретические знания о MongoDB, приятно удивила поддержка JavaScript, в частности ES5, что позволяет сократить время на написание кода, а также улучшит его читабельность.