Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«Работа с БД в СУБД MongoDB»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся (*Синюков Лев Владимирович)*
Факультет прикладной информатики
Группа __3240___
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

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.

Практическое задание 2.1.1:

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.*

```
test> use learn
switched to db learn
learn> db.unicorns.insertMany([
...   {name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63},
...   {name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43},
...   {name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182},
...   {name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99},
...   {name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80},
...   {name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40},
...   {name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39},
...   {name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2},
...   {name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33},
...   {name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54},
...   {name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'}
... ])
...
{
  acknowledged: true,
  insertedIds: {
            '0': ObjectId('683633411b53c923746c4bd0'),
            '1': ObjectId('683633411b53c923746c4bd1'),
            '2': ObjectId('683633411b53c923746c4bd2'),
            '3': ObjectId('683633411b53c923746c4bd3'),
            '4': ObjectId('683633411b53c923746c4bd4'),
            '5': ObjectId('683633411b53c923746c4bd5'),
            '6': ObjectId('683633411b53c923746c4bd6'),
            '7': ObjectId('683633411b53c923746c4bd7'),
            '8': ObjectId('683633411b53c923746c4bd8'),
            '9': ObjectId('683633411b53c923746c4bd9'),
            '10': ObjectId('683633411b53c923746c4bda')
  }
}
learn> let document = {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
... db.unicorns.insertOne(document)
...
{
  acknowledged: true,
  insertedId: ObjectId('683633461b53c923746c4bdb')
}
learn> db.unicorns.find().pretty()
[
  {
            _id: ObjectId('683633411b53c923746c4bd0'),
            name: 'Horny',
            loves: [ 'carrot', 'papaya' ],
            weight: 600,
            gender: 'm',
            vampires: 63
  },
```

```
{
        _id: ObjectId('683633411b53c923746c4bd1'),
        name: 'Aurora',
        loves: [ 'carrot', 'grape' ],
        weight: 450,
        gender: 'f',
        vampires: 43
},
{
        _id: ObjectId('683633411b53c923746c4bd2'),
        name: 'Unicrom',
        loves: [ 'energon', 'redbull' ],
        weight: 984,
        gender: 'm',
        vampires: 182
},
{
        _id: ObjectId('683633411b53c923746c4bd3'),
        name: 'Roooooodles',
        loves: [ 'apple' ],
        weight: 575,
        gender: 'm',
        vampires: 99
},
{
        _id: ObjectId('683633411b53c923746c4bd4'),
        name: 'Solnara',
        loves: [ 'apple', 'carrot', 'chocolate' ],
        weight: 550,
        gender: 'f',
        vampires: 80
},
{
        _id: ObjectId('683633411b53c923746c4bd5'),
        name: 'Ayna',
        loves: [ 'strawberry', 'lemon' ],
        weight: 733,
        gender: 'f',
        vampires: 40
},
{
        _id: ObjectId('683633411b53c923746c4bd6'),
        name: 'Kenny',
        loves: [ 'grape', 'lemon' ],
        weight: 690,
        gender: 'm',
        vampires: 39
},
{
        _id: ObjectId('683633411b53c923746c4bd7'),
        name: 'Raleigh',
        loves: [ 'apple', 'sugar' ],
        weight: 421,
        gender: 'm',
        vampires: 2
},
{
        _id: ObjectId('683633411b53c923746c4bd8'),
        name: 'Leia',
        loves: [ 'apple', 'watermelon' ],
        weight: 601,
        gender: 'f',
        vampires: 33
},
{
```

```
    _id: ObjectId('683633411b53c923746c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('683633411b53c923746c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('683633461b53c923746c4bdb'),
    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.

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

```
learn> db.unicorns.find({ gender: "f" })
[
  {
    _id: ObjectId('683633411b53c923746c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683633411b53c923746c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683633411b53c923746c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('683633411b53c923746c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
```

```
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('683633411b53c923746c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]

learn> db.unicorns.find({ name: "Aurora" })
[
  {
    _id: ObjectId('683633411b53c923746c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]


learn> db.unicorns.find({ name: { $ne: "Aurora" } })
[
  {
    _id: ObjectId('683633411b53c923746c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('683633411b53c923746c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683633411b53c923746c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('683633411b53c923746c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683633411b53c923746c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
```

```
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('683633411b53c923746c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('683633411b53c923746c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('683633411b53c923746c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('683633411b53c923746c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('683633411b53c923746c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('683633461b53c923746c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]

learn> db.unicorns.find({ weight: { $gt: 700 } })
[
  {
    _id: ObjectId('683633411b53c923746c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683633411b53c923746c4bd5'),
    name: 'Ayna',
```

```
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('683633461b53c923746c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]

learn> db.unicorns.find({ weight: { $lt: 500 } })
[
  {
    _id: ObjectId('683633411b53c923746c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683633411b53c923746c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]

learn> db.unicorns.find({ loves: "apple" })
[
  {
    _id: ObjectId('683633411b53c923746c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('683633411b53c923746c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683633411b53c923746c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('683633411b53c923746c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
```

```
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('683633411b53c923746c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
]

learn> db.unicorns.find({ loves: { $all: ["apple", "carrot"] } })
[
  {
    _id: ObjectId('683633411b53c923746c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
]
learn> db.unicorns.find({ gender: "f", weight: { $lt: 700 } })
[
  {
    _id: ObjectId('683633411b53c923746c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683633411b53c923746c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683633411b53c923746c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('683633411b53c923746c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

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

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({ $natural: -1 })
[
  {
          _id: ObjectId('683633461b53c923746c4bdb'),
          name: 'Dunx',
          loves: [ 'grape', 'watermelon' ],
          weight: 704,
          gender: 'm',
          vampires: 165
  },
  {
          _id: ObjectId('683633411b53c923746c4bda'),
          name: 'Nimue',
          loves: [ 'grape', 'carrot' ],
          weight: 540,
          gender: 'f'
  },
  {
          _id: ObjectId('683633411b53c923746c4bd9'),
          name: 'Pilot',
          loves: [ 'apple', 'watermelon' ],
          weight: 650,
          gender: 'm',
          vampires: 54
  },
  {
          _id: ObjectId('683633411b53c923746c4bd8'),
          name: 'Leia',
          loves: [ 'apple', 'watermelon' ],
          weight: 601,
          gender: 'f',
          vampires: 33
  },
  {
          _id: ObjectId('683633411b53c923746c4bd7'),
          name: 'Raleigh',
          loves: [ 'apple', 'sugar' ],
          weight: 421,
          gender: 'm',
          vampires: 2
  },
  {
          _id: ObjectId('683633411b53c923746c4bd6'),
          name: 'Kenny',
```

```
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('683633411b53c923746c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('683633411b53c923746c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683633411b53c923746c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('683633411b53c923746c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683633411b53c923746c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683633411b53c923746c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
]
```

Практическое задание 2.2.4:

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

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

```
        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',
        vampires: 182
},
{
        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
},
{
        name: 'Kenny',
        loves: [ 'grape' ],
        weight: 690,
        gender: 'm',
        vampires: 39
},
{
        name: 'Raleigh',
        loves: [ 'apple' ],
        weight: 421,
        gender: 'm',
        vampires: 2
},
{
        name: 'Leia',
        loves: [ 'apple' ],
        weight: 601,
        gender: 'f',
        vampires: 33
},
{
        name: 'Pilot',
        loves: [ 'apple' ],
```

```
        weight: 650,
        gender: 'm',
        vampires: 54
 },
 { name: 'Nimue', loves: [ 'grape' ], weight: 540, gender: 'f' },
 {
        name: 'Dunx',
        loves: [ 'grape' ],
        weight: 704,
        gender: 'm',
        vampires: 165
 }
]
```

Практическое задание 2.3.1:

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

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

Практическое задание 2.3.2:

Вывести список самцов единорогов

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

```
        name: 'Kenny',
        loves: [ 'grape', 'lemon' ],
        weight: 690,
        gender: 'm',
        vampires: 39
 }
]
```

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ `vampires`.

```
learn> db.unicorns.find(
...   { vampires: { $exists: false } }
... )
...
[
 {
        _id: ObjectId('683633411b53c923746c4bda'),
        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.

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

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"}}
```

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.*

```
learn> db.towns.insertMany([
...   {
...         name: "Punxsutawney",
...         populatiuon: 6200,
...         last_sensus: ISODate("2008-01-31"),
...         famous_for: [""],
...         mayor: {
...         name: "Jim Wehrle"
...         }
...   },
...   {
...         name: "New York",
...         populatiuon: 22200000,
...         last_sensus: ISODate("2009-07-31"),
...         famous_for: ["status of liberty", "food"],
...         mayor: {
...         name: "Michael Bloomberg",
...         party: "I"
...         }
...   },
...   {
...         name: "Portland",
...         populatiuon: 528000,
...         last_sensus: ISODate("2009-07-20"),
...         famous_for: ["beer", "food"],
...         mayor: {
...         name: "Sam Adams",
...         party: "D"
...         }
...   }
... ])
...
{
  acknowledged: true,
  insertedIds: {
          '0': ObjectId('6836394d1b53c923746c4bdc'),
          '1': ObjectId('6836394d1b53c923746c4bdd'),
          '2': ObjectId('6836394d1b53c923746c4bde')
  }
}
```

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

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

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.*
4) *Содержание коллекции единорогов* `unicorns`*:*

```
learn> var cursor = db.unicorns.find(
...   { gender: "m" }
... ).sort(
...   { name: 1 }
... ).limit(2)
...
... cursor.forEach(function(u) {
...   print(u.name + " — " + u.gender + ", вес: " + u.weight)
... })
...
Dunx — m, вес: 704
Horny — m, вес: 600
```

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 3.2.2:

Вывести список предпочтений.

```
learn> db.unicorns.aggregate([
...   { $unwind: "$loves" },
```

```
...   { $group: { _id: "$loves", count: { $sum: 1 } } },
...   { $sort: { count: -1 } }
... ])
...
[
  { _id: 'apple', count: 5 },
  { _id: 'grape', count: 4 },
  { _id: 'carrot', count: 4 },
  { _id: 'watermelon', count: 3 },
  { _id: 'lemon', count: 2 },
  { _id: 'papaya', count: 1 },
  { _id: 'strawberry', count: 1 },
  { _id: 'energon', count: 1 },
  { _id: 'sugar', count: 1 },
  { _id: 'redbull', count: 1 },
  { _id: 'chocolate', count: 1 }
]
```

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

```
learn> db.unicorns.countDocuments({ gender: "m" })
7
learn> db.unicorns.countDocuments({ gender: "f" })
5
```

Практическое задание 3.3.1:

1. *Выполнить команду:*

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

2. *Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.replaceOne(
...   { name: "Barny" },
...   {
...         name: "Barny",
...         loves: ["grape"],
...         weight: 340,
...         gender: "m"
...   },
...   { upsert: true }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Barny" }).pretty()
[
  {
          _id: ObjectId('68363d07c2f23e99d64342a6'),
          name: 'Barny',
          loves: [ 'grape' ],
          weight: 340,
          gender: 'm'
```

```
  }
]
```

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.updateOne(
...   { name: "Ayna", gender: "f" },
...   { $set: { weight: 800, vampires: 51 } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Ayna" }).pretty()
[
  {
          _id: ObjectId('683633411b53c923746c4bd5'),
          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.updateOne(
...   { name: "Raleigh", gender: "m" },
...   { $push: { loves: "redbull" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Raleigh" }).pretty()
[
  {
          _id: ObjectId('683633411b53c923746c4bd7'),
          name: 'Raleigh',
          loves: [ 'apple', 'sugar', '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: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
learn> db.unicorns.find({ gender: "m" }, { name: 1, vampires: 1, _id: 0 }).pretty()
[
  { name: 'Horny', vampires: 68 },
  { name: 'Unicrom', vampires: 187 },
  { name: 'Roooooodles', vampires: 104 },
  { name: 'Kenny', vampires: 44 },
  { name: 'Raleigh', vampires: 7 },
  { name: 'Pilot', vampires: 59 },
  { name: 'Dunx', vampires: 170 },
  { name: 'Barny', vampires: 5 }
]
```

Практическое задание 3.3.5:

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

```
learn> db.towns.updateOne(
...   { name: "Portland" },
...   { $unset: { "mayor.party": 1 } }
... )
... db.towns.find({ name: "Portland" }).pretty()
[
  {
          _id: ObjectId('6836394d1b53c923746c4bde'),
          name: 'Portland',
          populatiuon: 528000,
          last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
          famous_for: [ 'beer', 'food' ],
          mayor: { name: 'Sam Adams' }
  }
]
```

Практическое задание 3.3.6:

1. *Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.*
2. *Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.updateOne(
...   { name: "Pilot" },
...   { $push: { loves: "chocolate" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Pilot" }).pretty()
[
  {
          _id: ObjectId('683633411b53c923746c4bd9'),
          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.updateOne(
...   { name: "Aurora", gender: "f" },
...   { $addToSet: { loves: { $each: ["sugar", "lemon"] } } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Aurora" }).pretty()
[
  {
          _id: ObjectId('683633411b53c923746c4bd1'),
          name: 'Aurora',
          loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
          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.*
4) *Очистите коллекцию.*
5) *Просмотрите список доступных коллекций.*

```
learn> db.towns.find().pretty()
[
 {
        _id: ObjectId('6836394d1b53c923746c4bdc'),
        name: 'Punxsutawney',
        populatiuon: 6200,
        last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
        famous_for: [ '' ],
        mayor: { name: 'Jim Wehrle' }
 },
 {
        _id: ObjectId('6836394d1b53c923746c4bdd'),
        name: 'New York',
        populatiuon: 22200000,
        last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
        famous_for: [ 'status of liberty', 'food' ],
        mayor: { name: 'Michael Bloomberg', party: 'I' }
 },
 {
        _id: ObjectId('6836394d1b53c923746c4bde'),
        name: 'Portland',
        populatiuon: 528000,
        last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
        famous_for: [ 'beer', 'food' ],
        mayor: { name: 'Sam Adams' }
 },
 {
        _id: ObjectId('68363e981b53c923746c4bdf'),
        name: 'Punxsutawney',
        popujatiuon: 6200,
        last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
        famous_for: [ 'phil the groundhog' ],
        mayor: { name: 'Jim Wehrle' }
 },
 {
        _id: ObjectId('68363e981b53c923746c4be0'),
```

```
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('68363e981b53c923746c4be1'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]

learn> db.towns.remove({ "mayor.party": "I" })
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 2 }
learn> db.towns.find().pretty()
[
  {
    _id: ObjectId('6836394d1b53c923746c4bdc'),
    name: 'Punxsutawney',
    populatiuon: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('6836394d1b53c923746c4bde'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  },
  {
    _id: ObjectId('68363e981b53c923746c4bdf'),
    name: 'Punxsutawney',
    popujatiuon: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ 'phil the groundhog' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('68363e981b53c923746c4be1'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]

learn> db.towns.remove({})
{ acknowledged: true, deletedCount: 4 }
learn> show collections
towns
unicorns
```

Практическое задание 4.1.1:

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`*:*

```
learn> db.habitats.insertMany([
...   {
...         _id: "north_forest",
...         name: "Northern Forest",
...         description: "Cold coniferous forest with snowy peaks"
...   },
...   {
...         _id: "rain_valley",
...         name: "Rain Valley",
...         description: "Warm and humid rainforest valley"
...   }
... ])
...
{
  acknowledged: true,
  insertedIds: { '0': 'north_forest', '1': 'rain_valley' }
}

learn> db.unicorns.updateOne(
...   { name: "Aurora" },
...   { $set: { habitat_id: "north_forest" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> let habitatId = db.unicorns.findOne({ name: "Aurora" }).habitat_id
... db.habitats.findOne({ _id: habitatId })
...
{
  _id: 'north_forest',
  name: 'Northern Forest',
  description: 'Cold coniferous forest with snowy peaks'
}
```

Практическое задание 4.2.1:

1. *Проверьте, можно ли задать для коллекции* `unicorns` *индекс для ключа* `name` *с флагом* `unique`.
2. *Содержание коллекции единорогов* `unicorns`*:*

```
learn> db.unicorns.aggregate([
```

```
...   { $group: { _id: "$name", count: { $sum: 1 } } },
...   { $match: { count: { $gt: 1 } } }
... ])
...

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

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

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.dropIndex("name_1")
{ nIndexesWas: 2, ok: 1 }
learn> db.unicorns.dropIndex("_id_")
MongoServerError[InvalidOptions]: cannot drop _id index
```

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`*)*
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?*

```
learn> for (let i = 0; i < 100000; i++) {
...   db.numbers.insert({ value: i });
... }
...
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.

{
  acknowledged: true,
```

```
  insertedIds: { '0': ObjectId('683641411b53c923746dd281') }
}
learn>

learn> db.numbers
...   .find({})
...   .sort({ value: -1 })
...   .limit(4)
...   .explain("executionStats")
...
{
  explainVersion: '1',
  queryPlanner: {
            namespace: 'learn.numbers',
            parsedQuery: {},
            indexFilterSet: false,
            queryHash: 'BA27D965',
            planCacheShapeHash: 'BA27D965',
            planCacheKey: '7A892B81',
            optimizationTimeMillis: 0,
            maxIndexedOrSolutionsReached: false,
            maxIndexedAndSolutionsReached: false,
            maxScansToExplodeReached: false,
            prunedSimilarIndexes: false,
            winningPlan: {
            isCached: false,
            stage: 'SORT',
            sortPattern: { value: -1 },
            memLimit: 104857600,
            limitAmount: 4,
            type: 'simple',
            inputStage: { stage: 'COLLSCAN', direction: 'forward' }
            },
            rejectedPlans: []
  },
  executionStats: {
            executionSuccess: true,
            nReturned: 4,
            executionTimeMillis: 79,
            totalKeysExamined: 0,
            totalDocsExamined: 100000,
            executionStages: {
            isCached: false,
            stage: 'SORT',
            nReturned: 4,
            executionTimeMillisEstimate: 74,
            works: 100006,
            advanced: 4,
            needTime: 100001,
            needYield: 0,
            saveState: 4,
            restoreState: 4,
            isEOF: 1,
            sortPattern: { value: -1 },
            memLimit: 104857600,
            limitAmount: 4,
            type: 'simple',
            totalDataSizeSorted: 260,
            usedDisk: false,
            spills: 0,
            spilledDataStorageSize: 0,
            inputStage: {
            stage: 'COLLSCAN',
            nReturned: 100000,
            executionTimeMillisEstimate: 53,
```

```
            works: 100001,
            advanced: 100000,
            needTime: 0,
            needYield: 0,
            saveState: 4,
            restoreState: 4,
            isEOF: 1,
            direction: 'forward',
            docsExamined: 100000
            }
            }
 },
 queryShapeHash: 'A1C8CFCE9F916AB3A6ABCC8ABBB22506390F4D987582D3F926F0F2B36CA78396',
 command: {
            find: 'numbers',
            filter: {},
            sort: { value: -1 },
            limit: 4,
            '$db': 'learn'
 },
 serverInfo: {
            host: 'LB_SIN',
            port: 27017,
            version: '8.0.9',
            gitVersion: 'f882ef816d531ecfbb593843e4c554fda90ca416'
 },
 serverParameters: {
            internalQueryFacetBufferSizeBytes: 104857600,
            internalQueryFacetMaxOutputDocSizeBytes: 104857600,
            internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
            internalDocumentSourceGroupMaxMemoryBytes: 104857600,
            internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
            internalQueryProhibitBlockingMergeOnMongoS: 0,
            internalQueryMaxAddToSetBytes: 104857600,
            internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
            internalQueryFrameworkControl: 'trySbeRestricted',
            internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
 },
 ok: 1
}

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' }
]


learn> db.numbers
...   .find({})
...   .sort({ value: -1 })
...   .limit(4)
...   .explain("executionStats")
...
{
 explainVersion: '1',
 queryPlanner: {
            namespace: 'learn.numbers',
            parsedQuery: {},
            indexFilterSet: false,
            queryHash: 'BA27D965',
            planCacheShapeHash: 'BA27D965',
            planCacheKey: '7A892B81',
```

```
            optimizationTimeMillis: 0,
            maxIndexedOrSolutionsReached: false,
            maxIndexedAndSolutionsReached: false,
            maxScansToExplodeReached: false,
            prunedSimilarIndexes: false,
            winningPlan: {
            isCached: false,
            stage: 'LIMIT',
            limitAmount: 4,
            inputStage: {
            stage: 'FETCH',
            inputStage: {
            stage: 'IXSCAN',
            keyPattern: { value: 1 },
            indexName: 'value_1',
            isMultiKey: false,
            multiKeyPaths: { value: [] },
            isUnique: false,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'backward',
            indexBounds: { value: [ '[MaxKey, MinKey]' ] }
            }
            }
            },
            rejectedPlans: []
    },
    executionStats: {
            executionSuccess: true,
            nReturned: 4,
            executionTimeMillis: 1,
            totalKeysExamined: 4,
            totalDocsExamined: 4,
            executionStages: {
            isCached: false,
            stage: 'LIMIT',
            nReturned: 4,
            executionTimeMillisEstimate: 0,
            works: 5,
            advanced: 4,
            needTime: 0,
            needYield: 0,
            saveState: 0,
            restoreState: 0,
            isEOF: 1,
            limitAmount: 4,
            inputStage: {
            stage: 'FETCH',
            nReturned: 4,
            executionTimeMillisEstimate: 0,
            works: 4,
            advanced: 4,
            needTime: 0,
            needYield: 0,
            saveState: 0,
            restoreState: 0,
            isEOF: 0,
            docsExamined: 4,
            alreadyHasObj: 0,
            inputStage: {
            stage: 'IXSCAN',
            nReturned: 4,
            executionTimeMillisEstimate: 0,
            works: 4,
```

```
            advanced: 4,
            needTime: 0,
            needYield: 0,
            saveState: 0,
            restoreState: 0,
            isEOF: 0,
            keyPattern: { value: 1 },
            indexName: 'value_1',
            isMultiKey: false,
            multiKeyPaths: { value: [] },
            isUnique: false,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'backward',
            indexBounds: { value: [ '[MaxKey, MinKey]' ] },
            keysExamined: 4,
            seeks: 1,
            dupsTested: 0,
            dupsDropped: 0
            }
            }
            }
  },
  queryShapeHash: 'A1C8CFCE9F916AB3A6ABCC8ABBB22506390F4D987582D3F926F0F2B36CA78396',
  command: {
            find: 'numbers',
            filter: {},
            sort: { value: -1 },
            limit: 4,
            '$db': 'learn'
  },
  serverInfo: {
            host: 'LB_SIN',
            port: 27017,
            version: '8.0.9',
            gitVersion: 'f882ef816d531ecfbb593843e4c554fda90ca416'
  },
  serverParameters: {
            internalQueryFacetBufferSizeBytes: 104857600,
            internalQueryFacetMaxOutputDocSizeBytes: 104857600,
            internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
            internalDocumentSourceGroupMaxMemoryBytes: 104857600,
            internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
            internalQueryProhibitBlockingMergeOnMongoS: 0,
            internalQueryMaxAddToSetBytes: 104857600,
            internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
            internalQueryFrameworkControl: 'trySbeRestricted',
            internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
  },
  ok: 1
}
```


Время выполнения запроса без индекса составило 79мс, а с индексом всего 1мс, что является большой разницей.

Выводы

При работе над лабораторной работой 6 я узнал, что такое MongoDB, в чем ее отличие от реляционных БД. Создал базу данных, познакомился с синтаксисом, выполнил множество различных запросов.