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: Полухин А.В.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 7.0.1 (текущая), MongoShell.

Выполнение

Задание 2.1.1:

1. Создайте базу данных learn.
2. Заполните коллекцию единорогов unicorns:

```
> show dbs
< admin    32.00 KiB
  config  108.00 KiB
  local    72.00 KiB
> use learn
< switched to db learn
> db.createCollection('unicorns')
< { ok: 1 }
```

```
> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
  db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
  db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
  db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
  db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
  db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
  db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
  db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
  db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
  db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
  db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65df8572e7fd2c4540f0d504')
    }
  }
```

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

```
> document = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
< {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
> db.unicorns.insert(document)
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65df860ee7fd2c4540f0d505')
    }
  }
```

4. Проверьте содержимое коллекции с помощью метода find

```
> db.unicorns.find()
< {
    _id: ObjectId('65df8572e7fd2c4540f0d4fa'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d4fb'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d4fc'),
    name: 'Unicrom',
    loves: [
      'energon',
      'redbull'
    ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d4fd'),
    name: 'Roooooodles',
    loves: [
      'apple'
    ],
    weight: 575,
    gender: 'm',
    vampires: 99
  }
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d4fe'),
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],
  weight: 550,
  gender: 'f',
  vampires: 80
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d4ff'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 733,
  gender: 'f',
  vampires: 40
}
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d500'),
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d501'),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d502'),
  name: 'Leia',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 601,
  gender: 'f',
  vampires: 33
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d503'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 650,
  gender: 'm',
  vampires: 54
}
```

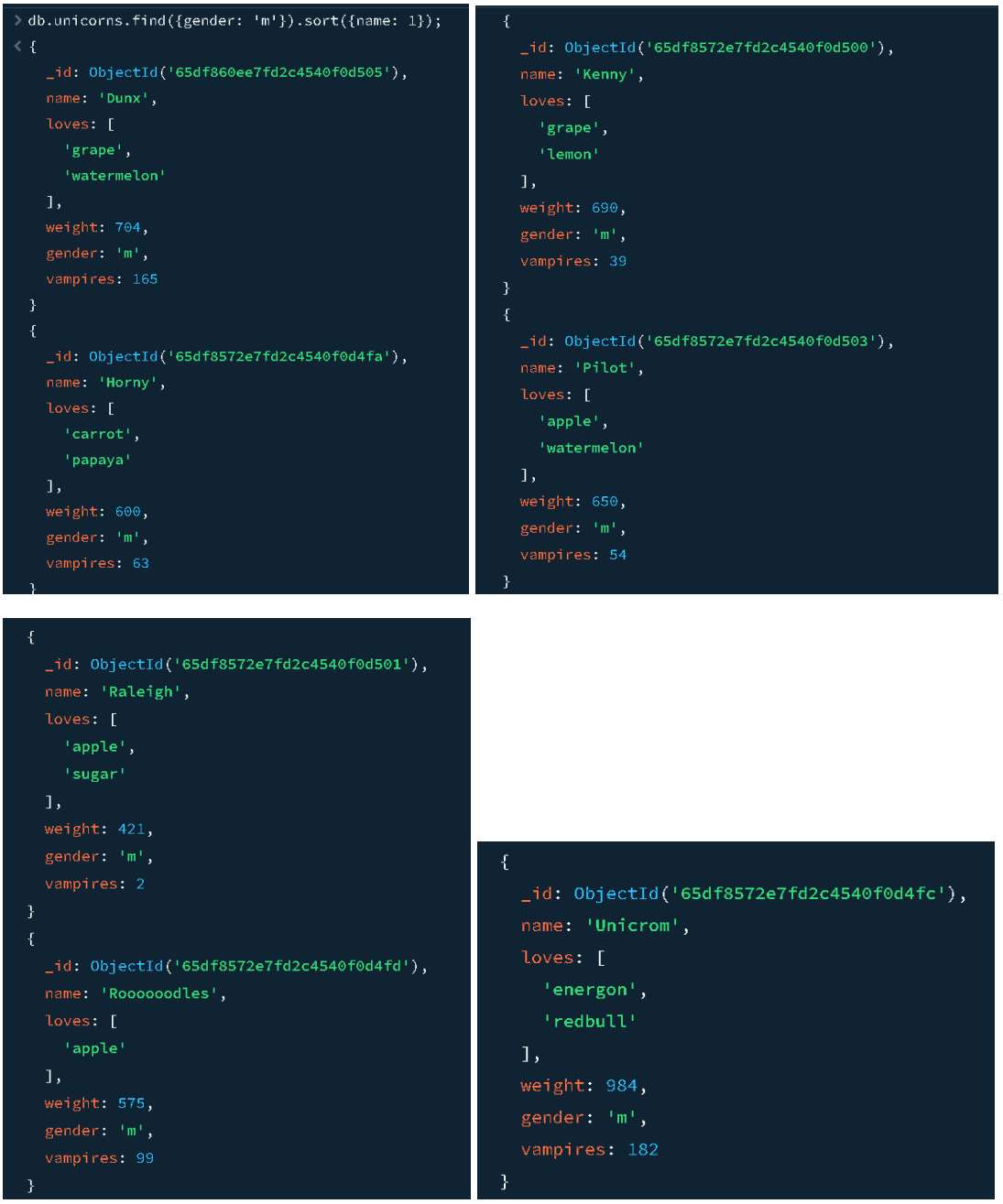
```
{
  _id: ObjectId('65df8572e7fd2c4540f0d504'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  _id: ObjectId('65df860ee7fd2c4540f0d505'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
```

Задание 2.2.1:

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

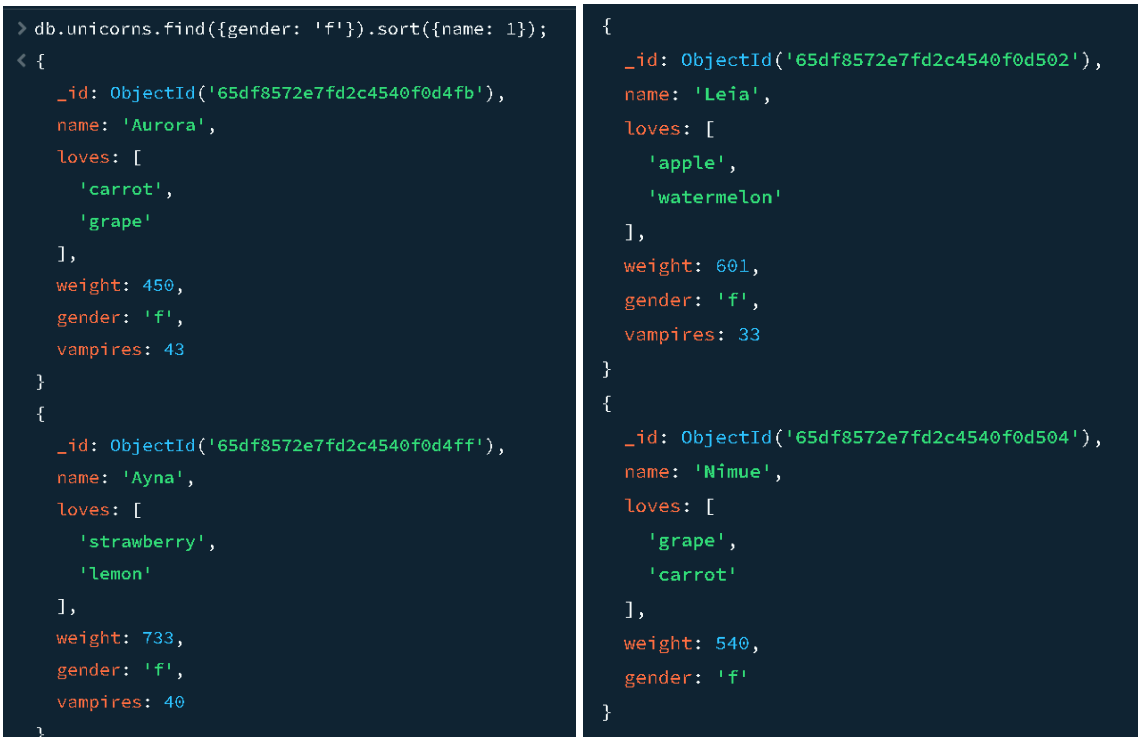
Листинг запроса для получения всех самцов:

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



Листинг запроса для получения всех самок:

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



```
> db.unicorns.find({gender: 'f'}).sort({name: 1});
< {
    _id: ObjectId('65df8572e7fd2c4540f0d4fb'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d4ff'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d502'),
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d504'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d4fe'),
    name: 'Solnara',
    loves: [
      'apple',
      'carrot',
      'chocolate'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
```

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

Листинг запроса с использованием limit:

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

```
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1);
< {
    _id: ObjectId('65df8572e7fd2c4540f0d4fb'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

Листинг запроса через findOne:

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

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'});
< {
    _id: ObjectId('65df8572e7fd2c4540f0d4fb'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

Задание 2.2.2:

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

Листинг запроса:

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

```
> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).sort({name: 1});
< {
    _id: ObjectId('65df860ee7fd2c4540f0d505'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d4fa'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d500'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d503'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d501'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d4fd'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d4fc'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
```

Задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

Листинг запроса:

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

```
> db.unicorns.find().sort({$natural: -1});
< {
    _id: ObjectId('65df860ee7fd2c4540f0d505'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d504'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d503'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 650,
  gender: 'm',
  vampires: 54
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d502'),
  name: 'Leia',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 601,
  gender: 'f',
  vampires: 33
}
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d501'),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d500'),
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d4ff'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 733,
  gender: 'f',
  vampires: 40
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d4fe'),
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],
  weight: 550,
  gender: 'f',
  vampires: 80
}
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d4fd'),
  name: 'Roooooodles',
  loves: [
    'apple'
  ],
  weight: 575,
  gender: 'm',
  vampires: 99
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d4fc'),
  name: 'Unicrom',
  loves: [
    'energon',
    'redbull'
  ],
  weight: 984,
  gender: 'm',
  vampires: 182
}
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d4fb'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d4fa'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```


Задание 2.1.4:

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

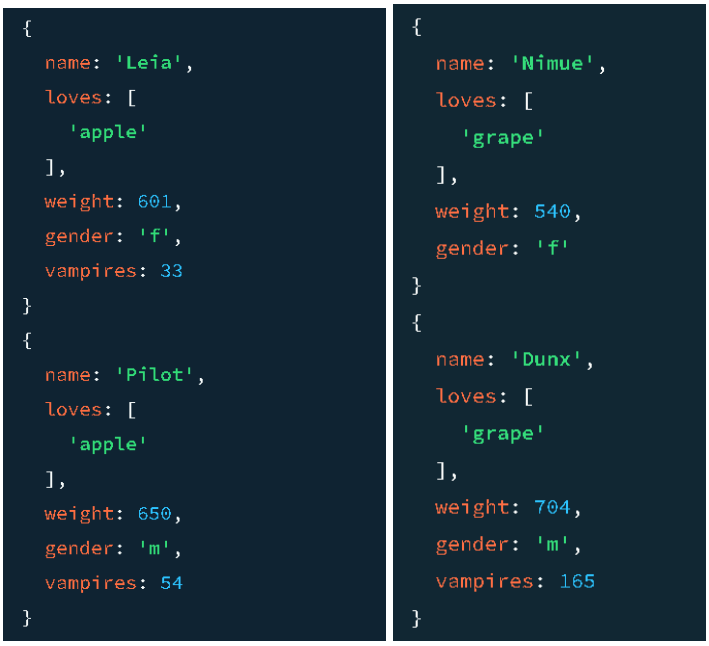
Листинг запроса:

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


```
> 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'
```

```
  name: 'Ayna',
  loves: [
    'strawberry'
  ],
  weight: 733,
  gender: 'f',
  vampires: 40
}
{
  name: 'Kenny',
  loves: [
    'grape'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
{
  name: 'Raleigh',
  loves: [
    'apple'
  ],
```



Задание 2.3.1:

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

Листинг запроса:

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

Задание 2.3.2:

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

Листинг запроса:

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

```
> db.unicorns.find({gender: 'm', weight: {$gte: 500}, loves: {$all: ['lemon', 'grape']}}, {_id: 0});
< {
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
```

Задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

Листинг запроса:

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

```
> db.unicorns.find({vampires: {exists: false}});
<
```

Задание 2.3.4:

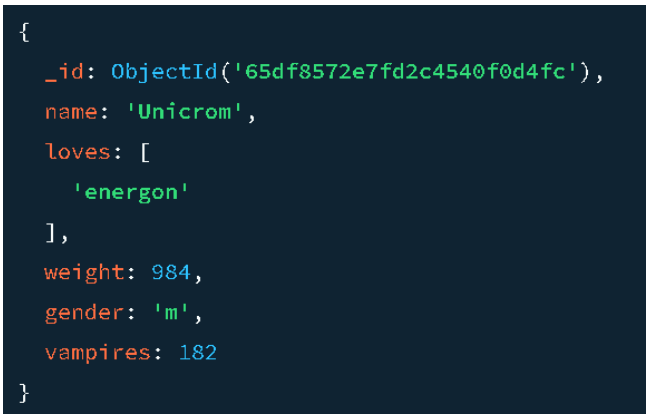
Вывести список упорядоченный список имен самцов единорогов с информацией об их первом предпочтении.

Листинг запроса:

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

```
> db.unicorns.find({gender: 'm'}, {loves: {$slice: 1}}).sort({name: 1});
< {
    _id: ObjectId('65df860ee7fd2c4540f0d505'),
    name: 'Dunx',
    loves: [
      'grape'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('65df8572e7fd2c4540f0d4fa'),
    name: 'Horny',
    loves: [
      'carrot'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d500'),
  name: 'Kenny',
  loves: [
    'grape'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d503'),
  name: 'Pilot',
  loves: [
    'apple'
  ],
  weight: 650,
  gender: 'm',
  vampires: 54
}
```



Задание 3.1.1:

- 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"}}
```

```
> db.createCollection('towns')
< { ok: 1 }
> 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('65df9055e7fd2c4540f0d507'),
      '1': ObjectId('65df9055e7fd2c4540f0d508'),
      '2': ObjectId('65df9055e7fd2c4540f0d509')
    }
  }
```

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

Листинг запроса:

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

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

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

Листинг запроса:

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

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

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.

Листинг запроса:

function printMaleUnicorns()

... {

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

... null;

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

... cursor.forEach(function(u)

... {print(u.name);}

...);

... }

```
> function printMaleUnicorns()
  {
  var cursor = db.unicorns.find({gender: 'm'});
  null;
  cursor.sort({name: 1}).limit(2);
  cursor.forEach(function(u)
  {print(u.name);}
  );
  }
< [Function: printMaleUnicorns]
> printMaleUnicorns()
< Dunx
< Horny
```

Задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

Листинг запроса:

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

```
> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 600}}).count()
< 2
```

Задание 3.2.2:

Вывести список предпочтений.

Листинг запроса:

db.unicorns.distinct('loves')

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

Задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

Листинг запроса:

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

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

Задание 3.3.1:

- Выполнить команду:

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

- Проверить содержимое коллекции `unicorns`.

```
{
  _id: ObjectId('65e0f18ae7fd2c4540f25baa'),
  name: 'Barny',
  loves: [
    'grape'
  ],
  weight: 340,
  gender: 'm'
}
```

Задание 3.3.2:

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

Листинг запроса:

db.unicorns.update({name: 'Ayna'}, {$set: {weight: 80, vampires: 51}})

```
> db.unicorns.update({name: 'Ayna'}, {$set: {weight: 80, vampires: 51}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

2. Проверить содержимое коллекции `unicorns`.

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d4ff'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 80,
  gender: 'f',
  vampires: 51
}
```

Задание 3.3.3:

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

Листинг запроса:

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

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

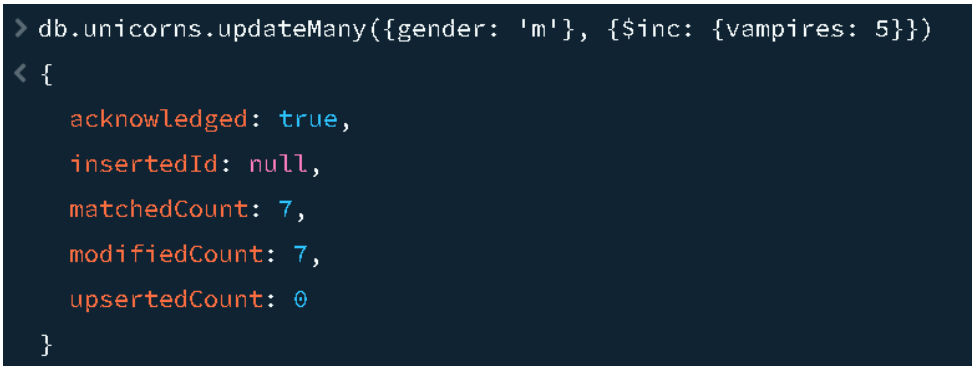
2. Проверить содержимое коллекции `unicorns`.

Задание 3.3.4:

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

Листинг запроса:

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



2. Проверить содержимое коллекции `unicorns`.

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d4fe'),
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],
  weight: 550,
  gender: 'f',
  vampires: 80
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d4ff'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 80,
  gender: 'f',
  vampires: 51
}
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d500'),
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 49
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d501'),
  name: 'Raleigh',
  loves: [
    'redbull'
  ],
  weight: 421,
  gender: 'm',
  vampires: 12
}
```

```
  _id: ObjectId('65df8572e7fd2c4540f0d502'),
  name: 'Leia',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 601,
  gender: 'f',
  vampires: 33
}
{
  _id: ObjectId('65df8572e7fd2c4540f0d503'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 64,
  area: DBRef('areas', 'fn')
}
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d504'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  _id: ObjectId('65df860ee7fd2c4540f0d505'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 175
}
```

```
{
  _id: ObjectId('65e0f18ae7fd2c4540f25baa'),
  name: 'Barny',
  loves: [
    'grape'
  ],
  weight: 340,
  gender: 'm'
}
```

Задание 3.3.5:

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

Листинг запроса:

db.towns.update({name: 'Portland'}, {$set: {'mayor.party': undefined}})

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

2. Проверить содержимое коллекции `towns`.

```
{
  _id: ObjectId('65df9055e7fd2c4540f0d509'),
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams',
    party: null
  }
}
```

Задание 3.3.6:

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

Листинг запроса:

db.unicorns.update({name: 'Pilot'}, {$push: {loves: 'chocolate'}})

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

2. Проверить содержимое коллекции `unicorns`.

```
> db.unicorns.find({name: 'Pilot'})
< {
    _id: ObjectId('65df8572e7fd2c4540f0d503'),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon',
      'chocolate'
    ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
```

Задание 3.3.7:

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

Листинг запроса:

db.unicorns.update({name: 'Aurora'}, {$push: {loves: {$each: ['sugar', 'lemons']}}})

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

2. Проверить содержимое коллекции `unicorns`.

```
> db.unicorns.find({name: 'Aurora'})
< {
    _id: ObjectId('65df8572e7fd2c4540f0d4fb'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar',
      'lemons'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

Задание 3.4.1:

1. Создайте коллекцию towns, включающую следующие документы:
2. Удалите документы с беспартийными мэрами.

Листинг запроса:

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

```
> db.towns.deleteMany({'mayor.party': {$exists: false}})
< {
    acknowledged: true,
    deletedCount: 1
  }
```

- Проверьте содержание коллекции.

```
< {
    _id: ObjectId('65df9055e7fd2c4540f0d508'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [
      'status of liberty',
      'food'
    ],
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
  {
    _id: ObjectId('65df9055e7fd2c4540f0d509'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [
      'beer',
      'food'
    ],
```

- Очистите коллекцию.

Листинг запроса: *db.towns.deleteMany({})*

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

- Просмотрите список доступных коллекций.

```
> show collections
towns
unicorns
```

Задание 4.1.1:

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

Листинг запроса:

db.createCollection('areas')

db.areas.insert({_id: 'eq', name: 'Equestria', description: 'Satrting base location'})

db.areas.insert({_id: 'ef', name: 'Everfree Forest', description: 'Magic forest with colorfull plants'})

db.areas.insert({_id: 'fn', name: 'Frozen North', description: 'Snowy mountains'})

db.areas.insert({_id: 'sl', name: 'Saddle Lake', description: 'Big clean lake'})

```
> db.createCollection('areas')
{ ok: 1 }
> db.areas.insert({_id: 'eq', name: 'Equestria', description: 'Satrting base location'})
{
  acknowledged: true,
  insertedIds: {
    '0': 'eq'
  }
}
> db.areas.insert({_id: 'ef', name: 'Everfree Forest', description: 'Magic forest with colorfull plants'})
{
  acknowledged: true,
  insertedIds: {
    '0': 'ef'
  }
}
> db.areas.insert({_id: 'fn', name: 'Frozen North', description: 'Snowy mountains'})
{
  acknowledged: true,
  insertedIds: {
    '0': 'fn'
  }
}
```

```
> db.areas.insert({_id: 'sl', name: 'Saddle Lake', description: 'Big clean lake'})
{
  acknowledged: true,
  insertedIds: {
    '0': 'sl'
  }
}
```

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

Листинг запроса:

db.unicorns.update({name: 'Pilot'}, {$set: {area: {$ref: 'areas', $id: 'fn'}}})

db.unicorns.update({name: 'Anya'}, {$set: {area: {$ref: 'areas', $id: 'sl'}}})

```
> db.unicorns.update({name: 'Pilot'}, {$set: {area: {$ref: 'areas', $id: 'fn'}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.update({name: 'Anya'}, {$set: {area: {$ref: 'areas', $id: 'sl'}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
```

3. Проверьте содержание коллекции единорогов.

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d503'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 64,
  area: DBRef('areas', 'fn')
}
```

```
{
  _id: ObjectId('65df8572e7fd2c4540f0d4ff'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 80,
  gender: 'f',
  vampires: 51
}
```

Задание 4.2.1:

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

Листинг запроса:

db.unicorns.ensureIndex({'name': 1}, {'unique': true})

```
> db.unicorns.ensureIndex({'name': 1}, {'unique': true})
< [ 'name_1' ]
```

Задание 4.3.1:

1. Получите информацию о всех индексах коллекции `unicorns` .

Листинг запроса:

db.unicorns.getIndexes()

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

2. Удалите все индексы, кроме индекса для идентификатора.

Листинг запроса:

db.unicorns.dropIndexes()

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

3. Попытайтесь удалить индекс для идентификатора.

Листинг запроса:

db.unicorns.dropIndex('_id_')

```
> db.unicorns.dropIndex('_id_')
MongoServerError: cannot drop _id index
```

Задание 4.4.1:

- Создайте объемную коллекцию `numbers`, задействовав курсор:

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

```
> db.createCollection('numbers')
< { ok: 1 }
> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65df9a26e7fd2c4540f25ba9')
    }
  }
```

- Выберите последних четыре документа.

Листинг запроса:

db.numbers.find().sort({value: -1}).limit(4).explain('executionStats')

```
> db.numbers.find().sort({value: -1}).limit(4).explain('executionStats')
< {
```

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

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 198,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
  executionStages: {
```

- Создайте индекс для ключа `value`.

  Листинг запроса:

  db.numbers.ensureIndex({'value': 1}, {'unique': true})

```
> db.numbers.ensureIndex({'value': 1}, {'unique': true})
< [ 'value_1' ]
```

- Получите информацию о всех индексах коллекции `numbres`.

  Листинг запроса:

  db.numbers.getIndexes()

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

- Выполните запрос 2.

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 6,
  totalKeysExamined: 4,
  totalDocsExamined: 4,
  executionStages: {
    stage: 'LIMIT',
```

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

В первом случае (без использования индексов) для выполнения запроса потребовалось 198 миллисекунд. Во втором случае (с использованием индексов) потребовалось 6 миллисекунды.

Вывод: в ходе лабораторной работы овладели 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.