

Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

ЛАБОРАТОРНАЯ РАБОТА №5 **«Работа с БД в СУБД MongoDB»**

Выполнил:
студент : Аль-Мошки Исмаил
Абдулвахаб
группа: К32401

Проверили:
Говорова Марина Михайловна

Санкт-Петербург
2023

- Цель:** овладеть практическими навыками работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.
- Оборудование:** компьютерный класс.
- Программное обеспечение:** СУБД MongoDB 4+, 6.0.6 (текущая).

Практическое задание 8.1.1:

1. Создайте базу данных *learn*:

Use learn.

2. Заполните коллекцию единорогов *unicorns*:

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

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

4. Проверьте содержимое коллекции с помощью метода *find*.

```
db.unicorns.find({name:"Dunx"})
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
```

```
learn> db.unicorns.insertOne({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
[{"_id": ObjectId("64749d57aebaed5df7182ac8"), "name": "Dunx", "loves": ["grape", "watermelon"], "weight": 704, "gender": "m", "vampires": 165}, {"_id": ObjectId("64749d8daebaed5df7182ac9"), "name": "Dunx", "loves": ["grape", "watermelon"], "weight": 704, "gender": "m", "vampires": 165}]
learn>
```

Практическое задание 8.1.2:

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

To get males db.unicorns.find({gender:"m"})

```
]
learn> db.unicorns.find({gender:"m"})
[
  {
    _id: ObjectId("64748709aebaed5df7182abd"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("64748709aebaed5df7182abf"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("64748709aebaed5df7182ac0"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("64748709aebaed5df7182ac3"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId("6474870aaebaed5df7182ac4"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId("6474870aaebaed5df7182ac6"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
```

To get Females db.unicorns.find({gender:"f"}).limit(1).sort({name:1})

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — □

learn> db.unicorns.find({gender:"f"}).limit(1).sort({name:1})
[
  {
    _id: ObjectId("64748709aebaed5df7182abe"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

To get them together `db.unicorns.find({$or:[{gender:"m"},{gender:"f"}]})`

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

```
db.unicorns.findOne({loves:"carrot"})
db.unicorns.find({loves:"carrot"}).limit(1)
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — □

learn> db.unicorns.find({loves:"carrot"}).limit(1)
[
  {
    _id: ObjectId("64748709aebaed5df7182abd"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
]
learn>
```

Практическое задание 8.1.3:

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

```
db.unicorns.find({gender:"m"},{loves:0,gender:0})
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
earn> db.unicorns.find({gender:"m"},{loves:0,gender:0})

{
  _id: ObjectId("64748709aebaed5df7182abd"),
  name: 'Horny',
  weight: 600,
  vampires: 63
},
{
  _id: ObjectId("64748709aebaed5df7182abf"),
  name: 'Unicrom',
  weight: 984,
  vampires: 182
},
{
  _id: ObjectId("64748709aebaed5df7182ac0"),
  name: 'Roooooodles',
  weight: 575,
  vampires: 99
},
{
  _id: ObjectId("64748709aebaed5df7182ac3"),
  name: 'Kenny',
  weight: 690,
  vampires: 39
},
{
  _id: ObjectId("6474870aaebaed5df7182ac4"),
  name: 'Raleigh',
  weight: 421,
  vampires: 2
},
{
  _id: ObjectId("6474870aaebaed5df7182ac6"),
  name: 'Pilot',
  weight: 650,
  vampires: 54
},
{
  _id: ObjectId("64749d57aebaed5df7182ac8"),
  name: 'Dunx',
  weight: 704,
  vampires: 165
},
{
  _id: ObjectId("64749d8daebaed5df7182ac9"),
  name: 'Dunx',
  weight: 704,
  vampires: 165
}

earn>
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

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

```
[mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000] learn> db.unicorns.find().sort({ $natural: -1 })  
[  
  {  
    _id: ObjectId("64749d8daebaed5df7182ac9"),  
    name: 'Dunx',  
    loves: [ 'grape', 'watermelon' ],  
    weight: 704,  
    gender: 'm',  
    vampires: 165  
  },  
  {  
    _id: ObjectId("64749d57aebaed5df7182ac8"),  
    name: 'Dunx',  
    loves: [ 'grape', 'watermelon' ],  
    weight: 704,  
    gender: 'm',  
    vampires: 165  
  },  
  {  
    _id: ObjectId("64748713aebaed5df7182ac7"),  
    name: 'Nimue',  
    loves: [ 'grape', 'carrot' ],  
    weight: 540,  
    gender: 'f'  
  },  
  {  
    _id: ObjectId("6474870aaebaed5df7182ac6"),  
    name: 'Pilot',  
    loves: [ 'apple', 'watermelon' ],  
    weight: 650,  
    gender: 'm',  
    vampires: 54  
  },  
  {  
    _id: ObjectId("6474870aaebaed5df7182ac5"),  
    name: 'Leia',  
    loves: [ 'apple', 'watermelon' ],  
    weight: 601,  
    gender: 'f',  
    vampires: 33  
  },  
  {  
    _id: ObjectId("6474870aaebaed5df7182ac4"),  
    name: 'Raleigh',  
    loves: [ 'apple', 'sugar' ],  
    weight: 421,  
    gender: 'm',  
    vampires: 2  
  },  
  {  
    _id: ObjectId("64748709aebaed5df7182ac3"),  
    name: 'Kenny',  
    loves: [ 'apple', 'watermelon' ],  
    weight: 540,  
    gender: 'f',  
    vampires: 33  
  }]
```

Практическое задание 8.1.5:

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

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



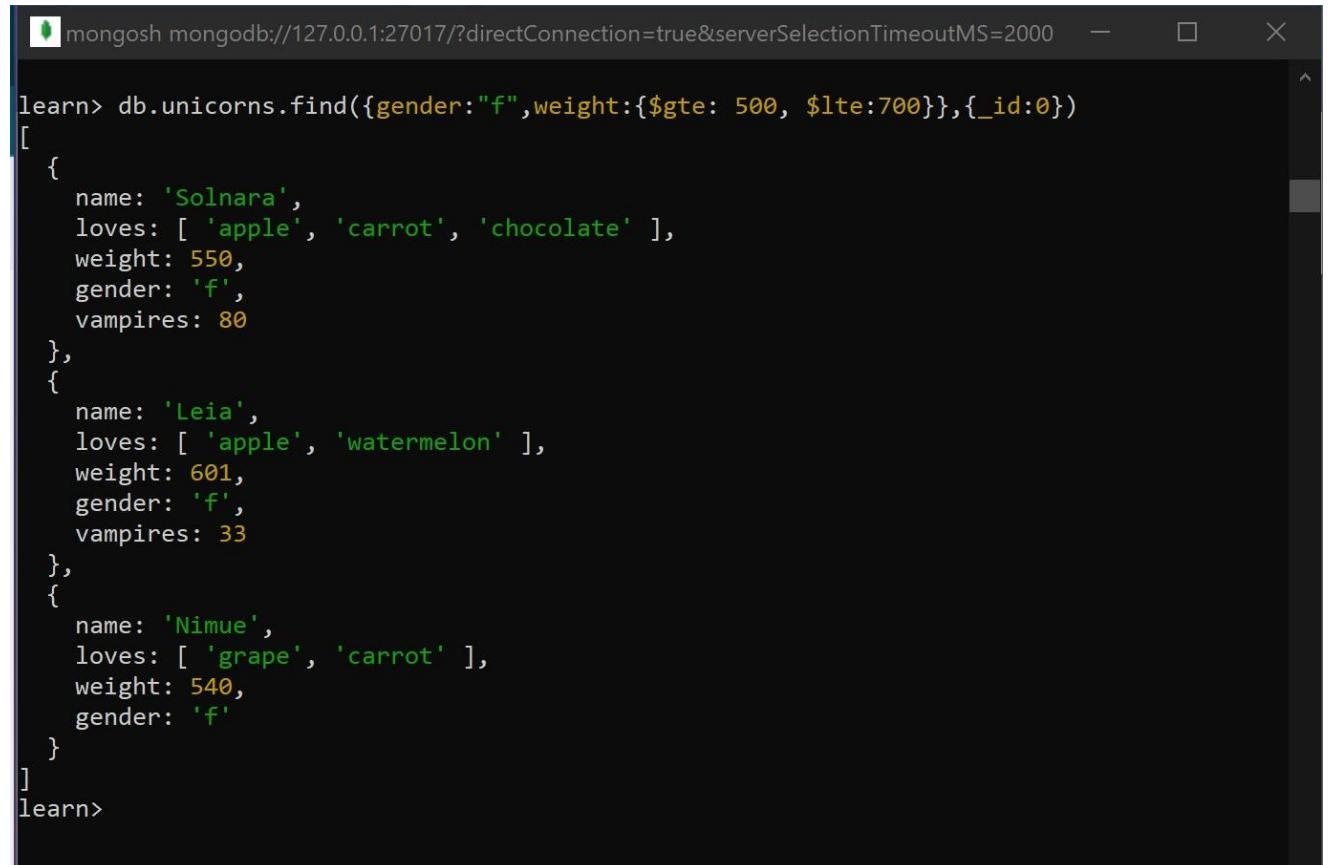
The screenshot shows a terminal window titled "mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000". The command entered is "db.unicorns.find({}, {_id:0, loves:{\$slice:1}})". The output displays a list of seven unicorn documents, each with its "_id" field removed and only the first item from the "loves" array retained. The "loves" array contains strings representing food items: 'carrot', 'energon', 'apple', 'strawberry', 'grape', 'milk', and 'icecream'.

```
learn> db.unicorns.find({}, {_id:0, loves:{$slice:1}})
[
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Aurora',
    loves: [ 'carrot' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    name: 'Unicrom',
    loves: [ 'energon' ],
    weight: 984,
    gender: 'm',
    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
  }
]
```

Практическое задание 8.1.6:

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

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



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
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'
  }
]
learn>
```

Практическое задание 8.1.7:

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

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



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
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
  }
]
learn>
```

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ vampires.

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

```
learn> db.unicorns.find({vampires:{$exists:false}})  
[  
  {  
    _id: ObjectId("64748713aebaed5df7182ac7"),  
    name: 'Nimue',  
    loves: [ 'grape', 'carrot' ],  
    weight: 540,  
    gender: 'f'  
  }  
]  
learn>
```

Практическое задание 8.1.9:

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

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — X

learn> db.unicorns.find({}, {_id:0, loves:{$slice:1}}).sort({name:1})
[
  {
    name: 'Aurora',
    loves: [ 'carrot' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    name: 'Ayna',
    loves: [ 'strawberry' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    name: 'Dunx',
    loves: [ 'grape' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    name: 'Dunx',
    loves: [ 'grape' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Kenny',
    loves: [ 'grape' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    name: 'Leia',
    loves: [ 'apple' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
]
```

8.2 ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB. ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ

1. ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ

Практическое задание 8.2.1:

1. Создайте коллекцию towns, включающую следующие документы

```
db.towns.insert({name:"Punxsutawney",populatiuon: 6200,last_sensus: ISODate("2008-01-31"),famous_for:[""],mayor:{name:"Jim Wehrle"}})
```

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

```
db.towns.insert({name: "Portland",populatiuon: 528000,last_sensus: ISODate("2009-07-20"),famous_for: ["beer", "food"],mayor: { name: "Sam Adams",party: "D"} })
```

```
learn> db.towns.insert({name:"Punxsutawney",populatiuon: 6200,last_sensus: ISODate("2008-01-31"),famous_for:[""],mayor:{name:"Jim Wehrle"}})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647503016c76df6c5520fce2") }
}
learn> db.towns.insert({name:"New York",populatiuon:22200000 ,last_sensus:ISODate("2009-07-31"),famous_for: ["status of liberty", "food"],mayor:{name: "Michael Bloomberg", ... party: "I"} })
...
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475039f6c76df6c5520fce3") }
}
learn> party: "I"
Uncaught:
SyntaxError: Unexpected token (1:10)

> 1 | party: "I"}}
      ^
  2 |

learn> db.towns.insert({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("6475041d6c76df6c5520fce4") }
}
learn>
```

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

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



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.towns.find({"mayor.party":"I"},{name:1,mayor:1,_id:0})
[
  {
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
learn>
```

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

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



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.towns.find({"mayor.party":{$exists:0}}, {name:1,mayor:1,_id:0})
[
  { name: 'Punxsutawney' },
  { name: 'Punxsutawney', mayor: { name: 'Jim Wehrle' } }
]
learn>
learn>
```

Практическое задание 8.2.2:

1. Сформировать функцию для вывода списка самцов единорогов.

```
db.users.find("this.gender=='m'")
```

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

```
var cursor = db.towns.find().limit(2).sort({name:1})
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — □

learn> var cursor = db.towns.find().limit(2).sort({name:1})

learn> cursor
[
  {
    _id: ObjectId("6475039f6c76df6c5520fce3"),
    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("6475041d6c76df6c5520fce4"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>
```

3. Вывести результат, используя *forEach*

```
cursor.forEach(function(obj){ print(obj.name); })
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — □

learn> var cursor = db.towns.find().limit(2).sort({name:1})

learn> cursor.forEach(function(obj){ print(obj.name); })
New York
Portland

learn>
```

4. Содержание коллекции единорогов *unicorns*:

```
db.unicorns.distinct("loves")
```

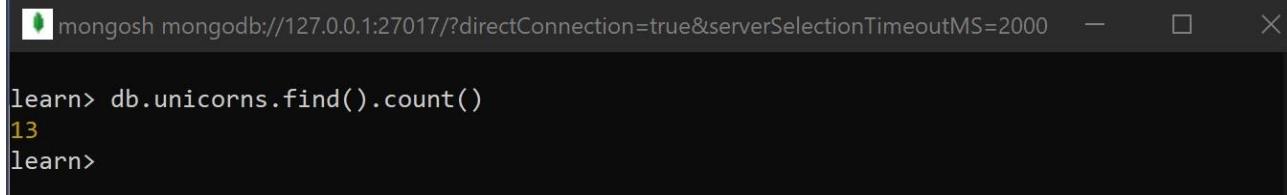
```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — □ X

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

Практическое задание 8.2.5:

Посчитать количество особей единорогов обоих полов.

```
db.unicorns.find().count()
```



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
```

```
learn> db.unicorns.find().count()
13
learn>
```

Практическое задание 8.2.6:

1. Выполнить команду:

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

2. Проверить содержимое коллекции unicorns.

Практическое задание 8.2.7:

1. Для самки единорога Ауны внести изменения в БД: теперь ее вес 800, она убила 51 вампира

```
db.unicorns.updateOne({name:"Ayna"},{$set:{weight:800, vampires:51}})
```



```
Select mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
```

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

2. Проверить содержимое коллекции unicorns.

```

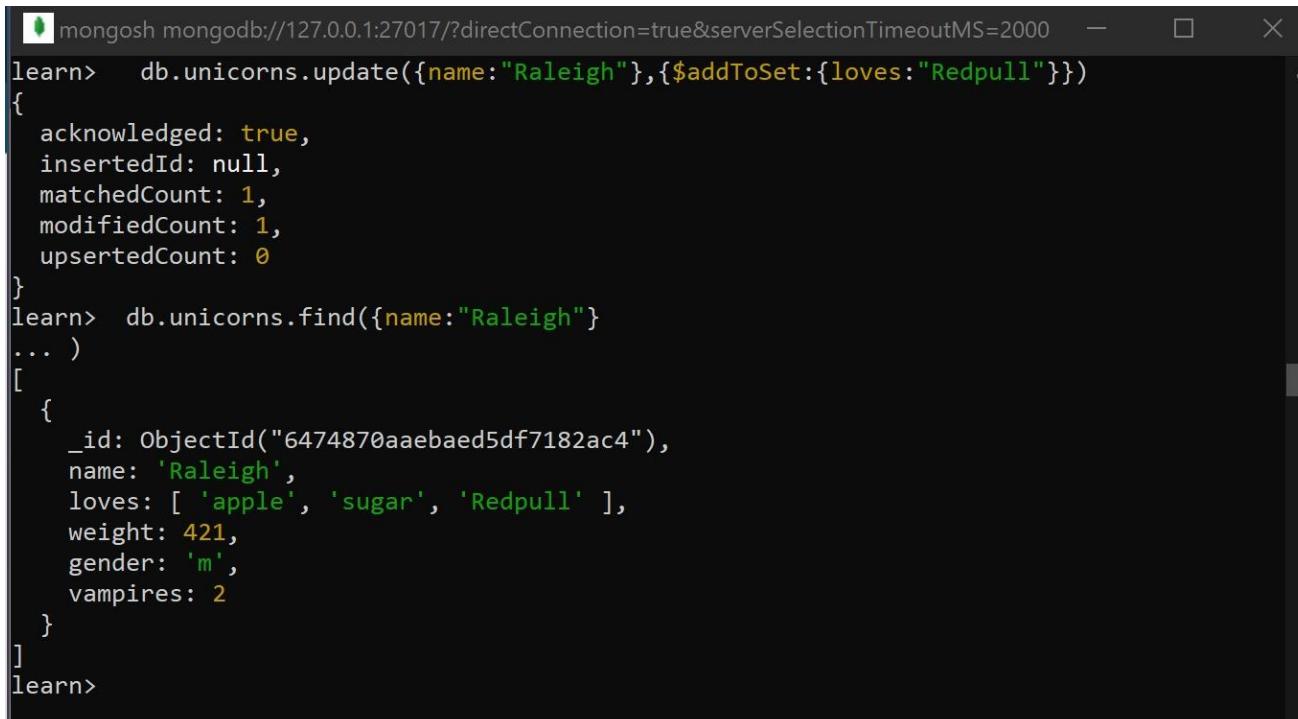
    },
    {
        _id: ObjectId("64748709aebaed5df7182ac1"),
        name: 'Solnara',
        loves: [ 'apple', 'carrot', 'chocolate' ],
        weight: 550,
        gender: 'f',
        vampires: 80
    },
    {
        _id: ObjectId("64748709aebaed5df7182ac2"),
        name: 'Ayna',
        loves: [ 'strawberry', 'lemon' ],
        weight: 800,
        gender: 'f',
        vampires: 51
    },
    {
        _id: ObjectId("64748709aebaed5df7182ac3"),
        name: 'Kenny',
        loves: [ 'grape', 'lemon' ],
        weight: 690,
        gender: 'm',
        vampires: 39
    },
}

```

Практическое задание 8.2.8:

1. Для самца единорога Raleigh внести изменения в БД: теперь он любит Redpull.
2. Проверить содержимое коллекции unicorns

```
db.unicorns.update({name:"Raleigh"},{$addToSet:{loves:"Redpull"}})
```



The screenshot shows the mongo shell interface. The command `db.unicorns.update({name:"Raleigh"},{\$addToSet:{loves:"Redpull"}})` is entered and executed. The response shows the updated document with the new love item added to the 'loves' array. The document is as follows:

```

{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
[ {
  _id: ObjectId("6474870aaebaed5df7182ac4"),
  name: 'Raleigh',
  loves: [ 'apple', 'sugar', 'Redpull' ],
  weight: 421,
  gender: 'm',
  vampires: 2
}]

```

Практическое задание 8.2.9:

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

Проверить содержимое коллекции `unicorns`.

```
db.unicorns.update({}, {$inc:{vampires:5}})
```

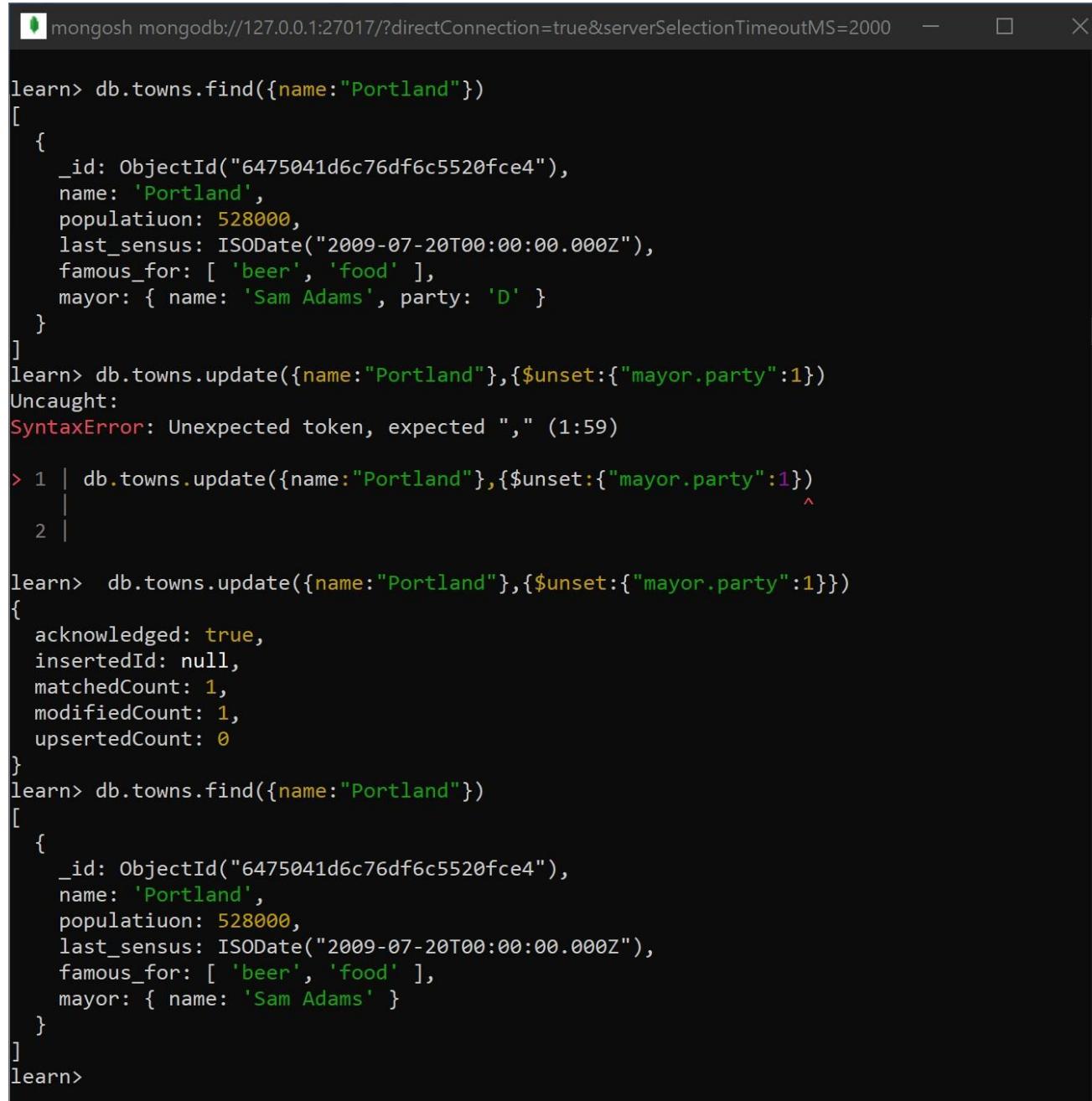
The screenshot shows a terminal window titled "mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000". The command `db.unicorns.update({}, {\$inc:{vampires:5}})` is run, followed by `db.unicorns.find()`. The output displays five documents from the `unicorns` collection, each with updated `vampires` values.

```
learn> db.unicorns.update({}, {$inc:{vampires:5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId("64748709aebaed5df7182abd"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId("64748709aebaed5df7182abe"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("64748709aebaed5df7182abf"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("64748709aebaed5df7182ac0"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("64748709aebaed5df7182ac1"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
]
```

Практическое задание 8.2.10:

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

```
db.towns.update({name:"Portland"},{$unset:{"mayor.party":1}})
```



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
```

```
learn> db.towns.find({name:"Portland"})
[{"_id": ObjectId("6475041d6c76df6c5520fce4"), "name": "Portland", "populatiuon": 528000, "last_sensus": ISODate("2009-07-20T00:00:00.000Z"), "famous_for": [ "beer", "food" ], "mayor": { "name": "Sam Adams", "party": "D" }}
```

```
learn> db.towns.update({name:"Portland"},{$unset:{"mayor.party":1}})
Uncaught:
SyntaxError: Unexpected token, expected "," (1:59)

> 1 | db.towns.update({name:"Portland"},{$unset:{"mayor.party":1}}) ^
2 |
```

```
learn> db.towns.update({name:"Portland"},{$unset:{"mayor.party":1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({name:"Portland"})
[{"_id": ObjectId("6475041d6c76df6c5520fce4"), "name": "Portland", "populatiuon": 528000, "last_sensus": ISODate("2009-07-20T00:00:00.000Z"), "famous_for": [ "beer", "food" ], "mayor": { "name": "Sam Adams" }}
```

```
learn>
```

Практическое задание 8.2.11:

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

Проверить содержимое коллекции unicorns.

```
db.unicorns.update({name:"Pilot"},{$push:{loves:"choklate"}})
[{"acknowledged": true,
 "insertedId": null,
 "matchedCount": 1,
 "modifiedCount": 1,
 "upsertedCount": 0}
]
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — ×
learn> db.unicorns.find({name:"Pilot"})
[
  {
    "_id": ObjectId("6474870aaebaed5df7182ac6"),
    "name": "Pilot",
    "loves": [ "apple", "watermelon", "choklate", "choklate" ],
    "weight": 650,
    "gender": "m",
    "vampires": 54,
    "love": [ "choklate" ]
  }
]
learn>
```

Практическое задание 8.2.12:

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

```
db.unicorns.update({name:"Aurora"},{$addToSet:{loves:{$each:["sugar","lemon"]}}})
```

```

learn> db.unicorns.update({name:"Aurora"},{$addToSet:{loves:$each:["sugar","lemon"]}})

{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:"Aurora"})
[
  {
    _id: ObjectId("64748709aebaed5df7182abe"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>

```



6. УДАЛЕНИЕ ДАННЫХ ИЗ КОЛЛЕКЦИИ

Практическое задание 8.2.13:

1. Создайте коллекцию towns, включающую следующие документы:

```

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

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

```

```
party: "I"}}

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

2. Удалите документы с беспартийными мэрами.
3. db.towns.remove({"mayor.party":{\$exists:false}},true)
4. Проверьте содержание коллекции

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — X ^  
learn> db.towns.find()  
[  
  {  
    _id: ObjectId("6475039f6c76df6c5520fce3"),  
    name: 'New York',  
    populatiuon: 2200000,  
    last_sensus: ISODate("2009-07-31T00:00:00.000Z"),  
    famous_for: [ 'status of liberty', 'food' ],  
    mayor: { name: 'Michael Bloomberg', party: 'I' }  
  },  
  {  
    _id: ObjectId("647553406c76df6c5520fce5"),  
    name: 'Punxsutawney ',  
    popujatiuon: 6200,  
    last_sensus: ISODate("2008-01-31T00:00:00.000Z")  
  },  
  {  
    _id: ObjectId("647554186c76df6c5520fce6"),  
    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": { $exists: false } }, true)  
{ acknowledged: true, deletedCount: 1 }  
learn> db.towns.find()  
[  
  {  
    _id: ObjectId("6475039f6c76df6c5520fce3"),  
    name: 'New York',  
    populatiuon: 2200000,  
    last_sensus: ISODate("2009-07-31T00:00:00.000Z"),  
    famous_for: [ 'status of liberty', 'food' ],  
    mayor: { name: 'Michael Bloomberg', party: 'I' }  
  },  
  {  
    _id: ObjectId("647554186c76df6c5520fce6"),  
    name: 'Portland',  
    popujatiuon: 528000,  
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),  
    famous_for: [ 'beer', 'food' ],  
    mayor: { name: 'Sam Adams', party: 'D' }  
  }  
]  
learn>
```

5. *Очистите коллекцию.*
6. *Просмотрите список доступных коллекций.*
db.towns.drop()



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.towns.drop()
true
learn> db.towns.find()
learn>
```

ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

ССЫЛКИ В БД

Практическое задание 8.3.1:

1. Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.
2. Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания.
3. Проверьте содержание коллекции единорогов.
4. Содержание коллекции единорогов `unicorns`:



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.update({name: "Kenny"}, {$set: {zone: {$ref: 'zones', $id: 'NY'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: "Kenny"})
[
  {
    _id: ObjectId("64748709aebaed5df7182ac3"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39,
    zone: DBRef("zones", 'NY')
  },
]
```

Практическое задание 8.3.2:

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

да, это возможно, поскольку в имени столбца в коллекции нет повторения значения

```
mongosh mongo://127.0.0.1:27017/?directConnection=true&serverSelectionTimeout...  
  insertedIds: { '0': ObjectId("647878968d91a3211840dbf0") }  
}  
learn> db.unicorns.insert({name: 'Roooooodles', dob: new Date(1979, 7, 18, 18, 44), love  
loves: ['apple'], weight: 575, gender: 'm', vampires: 99});  
{  
  acknowledged: true,  
  
learn> db.unicorns.ensureIndex({name:1},{unique:true})  
[ 'name_1' ]  
learn>
```

УПРАВЛЕНИЕ ИНДЕКСАМИ

Практическое задание 8.3.3:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 — X

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: cannot drop _id index
learn>
```

ПЛАН ЗАПРОСА

Практическое задание 8.3.4:

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

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

2. Выберите последних четыре документа.

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

4. Создайте индекс для ключа *value*.

5. Получите информацию о всех индексах коллекции *numbers*.

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?

Вывод:

в этой лабораторной работе я приобрел базовые навыки работы с NoSQL с использованием системы управления MongoDB.