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 Генне Константин Валерьевич
Факультет прикладной информатики
Группа K3240
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

1 Цель работы

– Овладеть практическими навыками установки СУБД MongoDB;

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

2 Практическое задание

В качестве предварительного знакомства с базовыми операциями MongoDB выполним практическое задание к лабораторной 6.1.

1) Проверьте работоспособность системы запуском клиента mongo.

```
C:\Users\K-Admin>mongosh
Current Mongosh Log ID: 68304b11b01c48db936c4bcf
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.1
Using MongoDB:          8.0.9
Using Mongosh:          2.5.1

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-05-22T23:50:30.827+03:00: Access control is not enabled for the database. Read and write access to data and configurati
on is unrestricted
------
```

2) Выполните методы

a) db.help()

```
test> db.help()

  Database Class:

    getMongo                                Returns the current database connection
    getName                                 Returns the name of the DB
    getCollectionNames                      Returns an array containing the names of all collections in the current database
.
    getCollectionInfos                      Returns an array of documents with collection information, i.e. collection name
and options, for the current database.
    runCommand                              Runs an arbitrary command on the database.
    adminCommand                            Runs an arbitrary command against the admin database.
```

b) db.help

```
test> db.help

  Database Class:

    getMongo                                Returns the current database connection
    getName                                 Returns the name of the DB
    getCollectionNames                      Returns an array containing the names of all collections in the current database
.
    getCollectionInfos                      Returns an array of documents with collection information, i.e. collection name
and options, for the current database.
    runCommand                              Runs an arbitrary command on the database.
    adminCommand                            Runs an arbitrary command against the admin database.
```

c) db.stats()

```
test> db.stats()
{
  db: 'test',
  collections: Long('0'),
  views: Long('0'),
  objects: Long('0'),
  avgObjSize: 0,
  dataSize: 0,
  storageSize: 0,
  indexes: Long('0'),
  indexSize: 0,
  totalSize: 0,
  scaleFactor: Long('1'),
  fsUsedSize: 0,
  fsTotalSize: 0,
  ok: 1
}
```

3) Создайте БД learn

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

4) Получите список доступных БД

```
learn> show dbs
admin   40.00 KiB
config  72.00 KiB
local   40.00 KiB
```

5) Создайте коллекцию unicorns, вставив в нее документ {name: 'Aurora', gender: 'f', weight: 450}.

```
learn> db.unicorns.insertOne({name: 'Aurora', gender: 'f', weight: 450})
{
  acknowledged: true,
  insertedId: ObjectId('68304ebab01c48db936c4bd1')
}
```

6) Просмотрите список текущих коллекций.

```
learn> show collections
unicorns
```

7) Переименуйте коллекцию unicorns.

```
learn> db.unicorns.renameCollection('super_unicorns')
{ ok: 1 }
```

8) Просмотрите статистику коллекции.

9) Удалите коллекцию.

```
learn> db.super_unicorns.drop()
true
```

10) Удалите БД learn

```
learn> db.dropDatabase()
{ ok: 1, dropped: 'learn' }
```

2.1 Вставка документов в коллекцию

Практическое задание 2.1.1

1) Создайте базу данных learn.

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

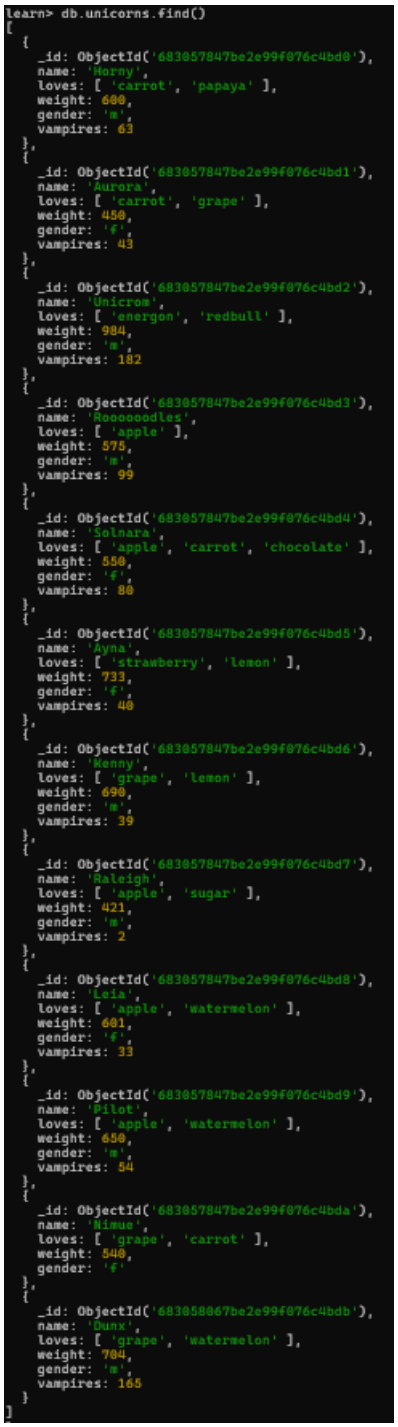
2) Заполните коллекцию единорогов unicorns: db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63}); db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43}); db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182}); db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99}); db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80}); db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40}); db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39}); db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2}); db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33}); db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54}); db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});

```
learn> 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('683057847be2e99f076c4bda') }
}
```

3) Используя второй способ, вставьте в коллекцию единорогов документ: document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})

```
learn> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683058067be2e99f076c4bdb') }
}
```

4) Проверьте содержимое коллекции с помощью метода find.



2.2 Выборка данных из БД

Практическое задание 2.2.1

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

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('683058067be2e99f076c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('683057847be2e99f076c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

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

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId('683057847be2e99f076c4bd1'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

```
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId('683057847be2e99f076c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 2.2.2

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).sort({name: 1})
[
  {
    _id: ObjectId('683058067be2e99f076c4bdb'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd0'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd6'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd9'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd7'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd3'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd2'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
]
```

Практическое задание 2.2.3

1) Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({ $natural: -1 })
[
  {
    _id: ObjectId('683058067be2e99f076c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('683057847be2e99f076c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd4'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
]
```

Практическое задание 2.2.4

1) 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
  }
]
```

2.3 Логические операторы

Практическое задание 2.3.1

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

```
learn> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lt: 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

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

```
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

1) Найти всех единорогов, не имеющих ключ vampires.

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('683057847be2e99f076c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.4

1) 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' ] }
]
```

3.1 Запрос к вложенным объектам

Практическое задание 3.1.1

1) Создайте коллекцию towns, включающую следующие документы:

{name: "Punxsutawney ",
populatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: [""],
mayor: {
  name: "Jim Wehrle"
}}
{name: "New York",
populatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
  name: "Michael Bloomberg",
party: "I"}}
{name: "Portland",
populatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
  name: "Sam Adams",
party: "D"}}

```
learn> db.towns.insertMany([
... {name: "Punxsutawney ",
... 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('68306daf7be2e99f076c4bdc'),
    '1': ObjectId('68306daf7be2e99f076c4bdd'),
    '2': ObjectId('68306daf7be2e99f076c4bde')
  }
}
```

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

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

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

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

Практическое задание 3.1.2

1) Сформировать функцию для вывода списка самцов единорогов.

```
learn> fn = function() { return this.gender == 'm'; }
[Function: fn]
```

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

```
learn> var cursor = db.unicorns.find({$where: fn}).sort({name: 1}).limit(2); null;
null
```

3) Вывести результат, используя forEach.

```
learn> cursor.forEach(function(obj){ print(obj.name); })
Dunx
Horny
```

3.2 Агрегированные запросы

Практическое задание 3.2.1

1) Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 3.2.2

1) Вывести список предпочтений.

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

Практическое задание 3.2.3

1) Подсчитать количество особей единорогов обоих полов.

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

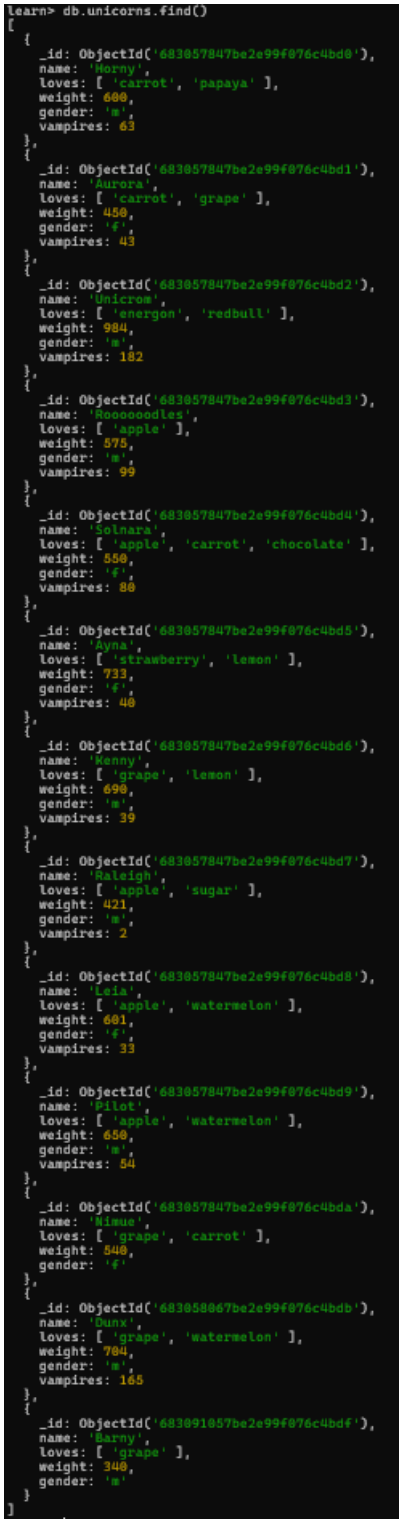
3.3 Редактирование данных

Практическое задание 3.3.1

1) Выполнить команду: db.unicorns.insertOne({ name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})

```
learn> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedId: ObjectId('683091057be2e99f076c4bdf')
}
```

2) Проверить содержимое коллекции unicorns.

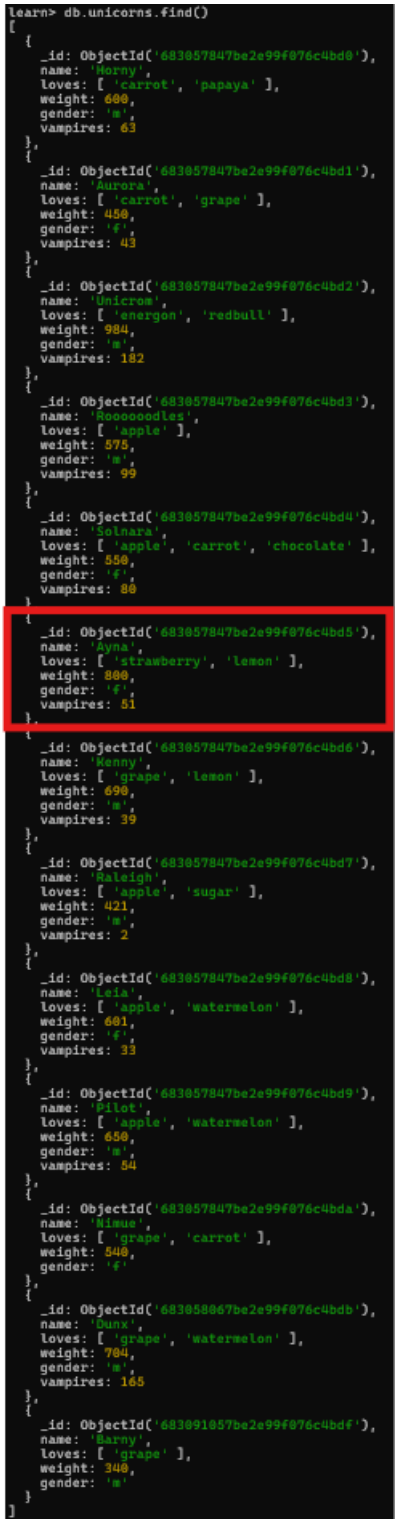


Практическое задание 3.3.2

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

```
learn> db.unicorns.update({gender: 'f', name: 'Ayna'}, {$set: {weight: 800, vampires: 51}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2) Проверить содержимое коллекции unicorns.



Практическое задание 3.3.3

1) Для самца единорога Raleigh внести изменения в БД: теперь он любит ред булл.

```
learn> db.unicorns.update({gender: 'm', name: 'Raleigh'}, {$set: {loves: ['redbull']}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2) Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.find({name: 'Raleigh'})
[
  {
    _id: ObjectId('683057847be2e99f076c4bd7'),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Практическое задание 3.3.4

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

```
learn> db.unicorns.update({gender: 'm'}, {$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2) Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.find({gender: 'm'})
[
  {
    _id: ObjectId('683057847be2e99f076c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd7'),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('683057847be2e99f076c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('683058067be2e99f076c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('683091057be2e99f076c4bdf'),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
```

Практическое задание 3.3.5

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

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

2) Проверить содержимое коллекции towns.

```
learn> db.towns.find({name: 'Portland'})
[
  {
    _id: ObjectId('68306daf7be2e99f076c4bde'),
    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.

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

2) Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.find({name: 'Pilot'})
[
  {
    _id: ObjectId('683057847be2e99f076c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
]
```

Практическое задание 3.3.7

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

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

2) Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.find({name: 'Aurora'})
[
  {
    _id: ObjectId('683057847be2e99f076c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

3.4 Удаление данных из коллекции

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"}}
```

```
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('6831f5b3f8d9d4ed6c6c4bd0'),
    '1': ObjectId('6831f5b3f8d9d4ed6c6c4bd1'),
    '2': ObjectId('6831f5b3f8d9d4ed6c6c4bd2')
  }
}
```

2) Удалите документы с беспартийными мэрами.

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

3) Проверьте содержание коллекции.

```
learn> db.towns.find()
[
  {
    _id: ObjectId('6831f5b3f8d9d4ed6c6c4bd1'),
    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('6831f5b3f8d9d4ed6c6c4bd2'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

4) Очистите коллекцию.

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

5) Просмотрите список доступных коллекций.

```
learn> show collections
towns
unicorns
```

4.1 Ссылки в БД

Практическое задание 4.1.1

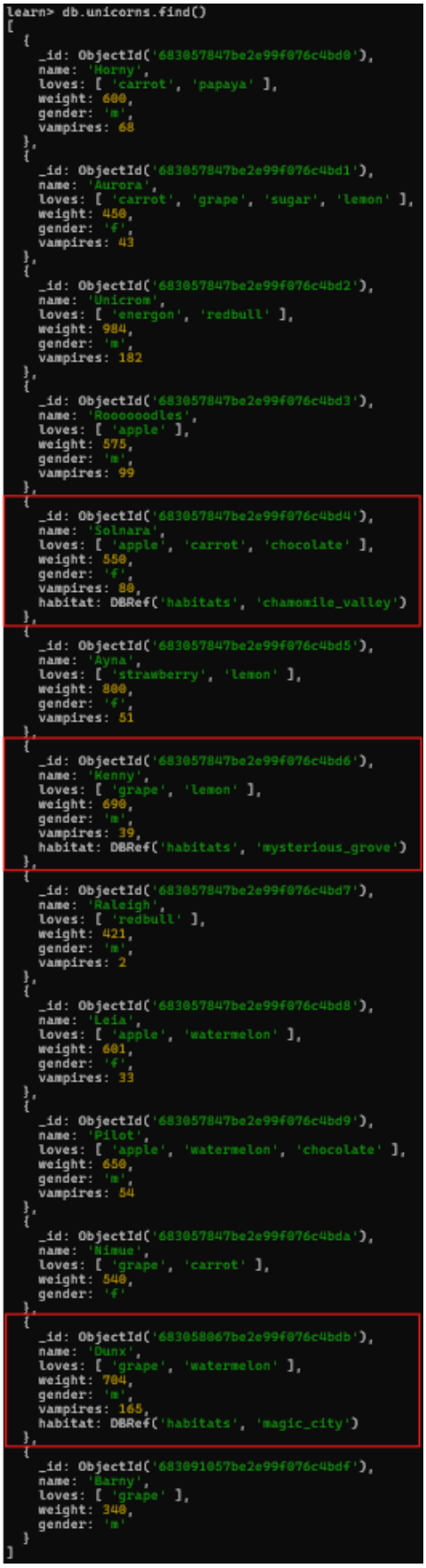
1) Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.

```
learn> db.habitats.insertMany([
... {_id: 'chamomile_valley', name: 'Chamomile Valley', description: 'Chamomile Valley is a place where there is no way for vampires to pass'},
... {_id: 'magic_city', name: 'Magical City', description: 'The city where dreams come true'},
... {_id: 'mysterious_grove', name: 'The mysterious grove', description: 'A real paradise on Earth'},
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': 'chamomile_valley',
    '1': 'magic_city',
    '2': 'mysterious_grove'
  }
}
```

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

```
learn> db.unicorns.updateOne({name: 'Dunx'}, {$set: {habitat: {$ref: 'habitats', $id: 'magic_city'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne({name: 'Kenny'}, {$set: {habitat: {$ref: 'habitats', $id: 'mysterious_grove'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne({name: 'Solnara'}, {$set: {habitat: {$ref: 'habitats', $id: 'chamomile_valley'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

3) Проверьте содержание коллекции единорогов.



4.2 Настройка индексов

Практическое задание 4.2.1

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

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

4.3 Управление индексами

Практическое задание 4.3.1

1) Получите информацию о всех индексах коллекции unicorns.

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

2) Удалите все индексы, кроме индекса для идентификатора.

```
learn> db.unicorns.dropIndex('name_1')
{ nIndexesWas: 2, ok: 1 }
```

3) Попытайтесь удалить индекс для идентификатора.

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

4.4 План запроса

Практическое задание 4.4.1

1) Создайте объемную коллекцию numbers, задействовав курсор: for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}

```
learn> for(i = 0; i < 100000; i++){db.numbers.insertOne({value: i})}
{
  acknowledged: true,
  insertedId: ObjectId('68324457f8d9d4ed6c6dd272')
}
```

2) Выберите последних четыре документа.

```
learn> db.numbers.find().sort({value: -1}).limit(4)
[
  { _id: ObjectId('68324457f8d9d4ed6c6dd272'), value: 99999 },
  { _id: ObjectId('68324457f8d9d4ed6c6dd271'), value: 99998 },
  { _id: ObjectId('68324457f8d9d4ed6c6dd270'), value: 99997 },
  { _id: ObjectId('68324457f8d9d4ed6c6dd26f'), value: 99996 }
]
```

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

```
learn> db.numbers.explain('executionStats').find().sort({value: -1}).limit(4)
{
  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: 55,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      isCached: false,
      stage: 'SORT',
```

Время на выполнение запроса – 55 миллисекунд.

4) Создайте индекс для ключа value.

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

5) Получите информацию о всех индексах коллекции numbers.

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

6) Выполните запрос 2.

```
learn> db.numbers.find().sort({value: -1}).limit(4)
[
  { _id: ObjectId('68324457f8d9d4ed6c6dd272'), value: 99999 },
  { _id: ObjectId('68324457f8d9d4ed6c6dd271'), value: 99998 },
  { _id: ObjectId('68324457f8d9d4ed6c6dd270'), value: 99997 },
  { _id: ObjectId('68324457f8d9d4ed6c6dd26f'), value: 99996 }
]
```

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

```
learn> db.numbers.explain('executionStats').find().sort({value: -1}).limit(4)
{
  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: true,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'backward',
          indexBounds: { value: [ '[MaxKey, MinKey]' ] }
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 2,
    totalKeysExamined: 4,
    totalDocsExamined: 4,
    executionStages: {
      isCached: false,
      stage: 'LIMIT',
```

С установленным индексом время на выполнение запроса составило 2 миллисекунды.

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

Время на выполнения запроса без индекса составило 55 миллисекунд, с индексом – 2 миллисекунды, что на приблизительно 96% меньше. Таким образом, запрос с индексом более эффективен.

3 Вывод

В ходе лабораторной работы я изучил и отработал на практике основные операции с нереляционной базой данных MongoDB. Были выполнены задачи по добавлению данных в коллекции, проведена фильтрация документов по полям с использованием сравнительных и логических операторов, применены проекции для выбора конкретных полей, реализована сортировка результатов запросов. Также были освоены операции обновления данных различными способами, удаления документов по заданным условиям, работы с вложенными структурами и массивами. Кроме того, я научился создавать и удалять индексы, изучил их влияние на скорость выполнения запросов. В результате проделанной работы я получил важный практический опыт взаимодействия с NoSQL базами данных.