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

Отчет

по лабораторной работе «Работа с БД в СУБД MongoDB»

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

Автор: Коротин А.М.

Факультет: ИКТ

Группа: К32391

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+, 6.0.6 (текущая).

Ход работы:

Для локального запуска СУБД MongoDB использовался Docker контейнер.

```yaml
version: '3.1'

services:

  mongo:
    image: mongo
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root
    volumes:
      - './volumes:/data/db'
```

docker-compose.yml

Практическое задание 8.1.1:

1) *Создайте базу данных learn.*

```
admin> db.auth('root', 'root')
{ ok: 1 }
admin> use learn
switched to db learn
learn>
```

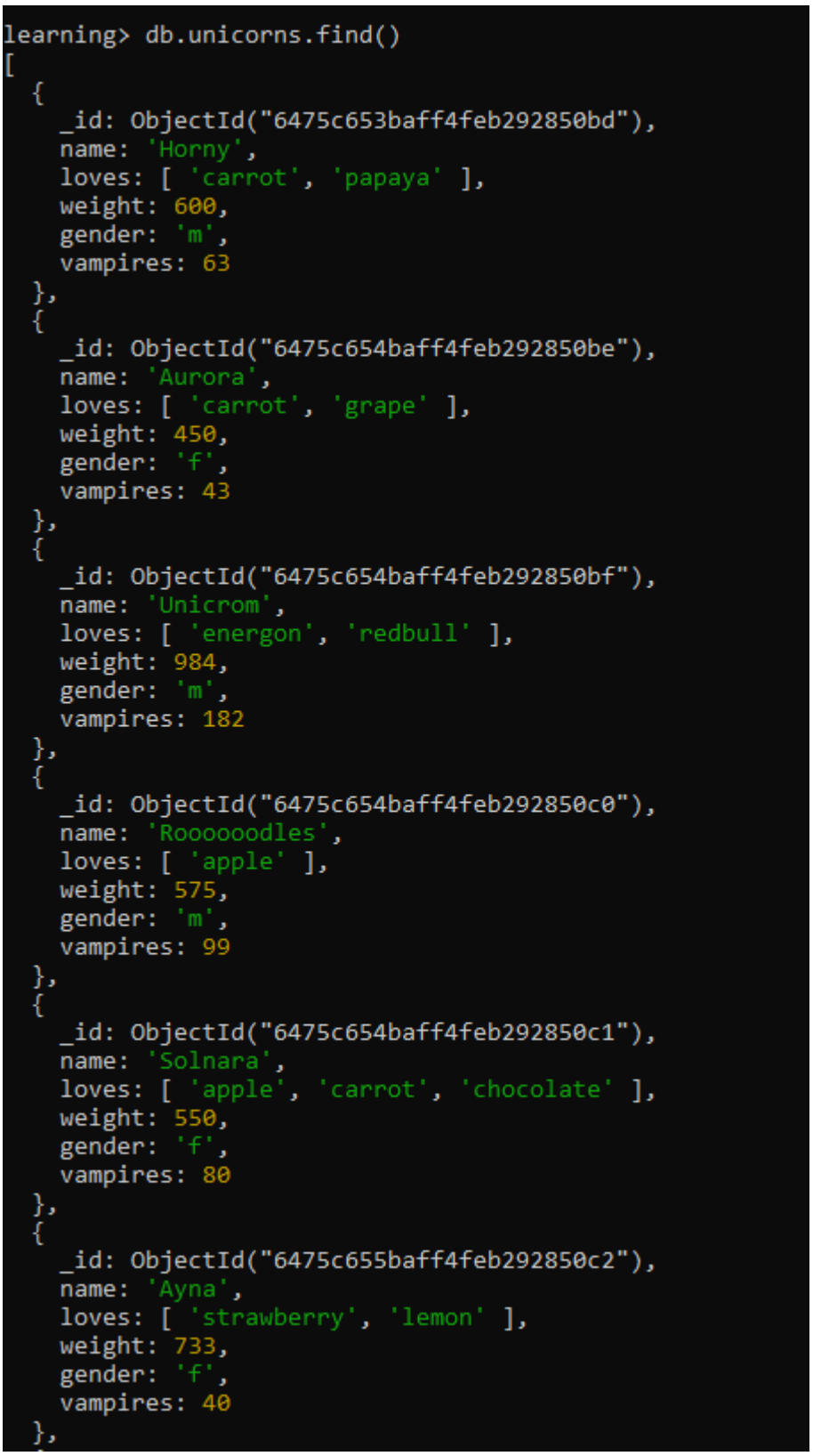
2) *Заполните коллекцию единорогов unicorns*

```
learning> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c653baff4feb292850bd") }
}
learning> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c654baff4feb292850be") }
}
learning> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c654baff4feb292850bf") }
}
learning> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c654baff4feb292850c0") }
}
learning> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c654baff4feb292850c1") }
}
learning> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c655baff4feb292850c2") }
}
learning> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c655baff4feb292850c3") }
}
learning> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c655baff4feb292850c4") }
}
learning> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c655baff4feb292850c5") }
}
learning> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
```

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

```
learning> doc = {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learning> db.unicorns.insert(doc)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6475c6aabaff4feb292850c8") }
}
```

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



Список объектов неполон, информация обо всех объектах коллекции не помещается в стандартный буфер терминала вывода.

Практическое задание 8.1.2:

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

```
learning> db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
[
  {
    _id: ObjectId("6475c654baff4feb292850be"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("6475c655baff4feb292850c2"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId("6475c655baff4feb292850c5"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

Список самок

```
learning> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId("6475c6aabaff4feb292850c8"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("6475c653baff4feb292850bd"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("6475c655baff4feb292850c3"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId("6475c655baff4feb292850c6"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId("6475c655baff4feb292850c4"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId("6475c654baff4feb292850c0"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("6475c654baff4feb292850bf"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

Список самцов

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

```
learning> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId("6475c654baff4feb292850be"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

```
learning> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId("6475c654baff4feb292850be"),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Практическое задание 8.1.3:

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

```
learning> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).sort({name: 1})
[
  {
    _id: ObjectId("6475c6aabaff4feb292850c8"),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId("6475c653baff4feb292850bd"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId("6475c655baff4feb292850c3"),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId("6475c655baff4feb292850c6"),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId("6475c655baff4feb292850c4"),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId("6475c654baff4feb292850c0"),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId("6475c654baff4feb292850bf"),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
]
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

```
learning> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId("6475c6aabaff4feb292850c8"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("6475c655baff4feb292850c7"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("6475c655baff4feb292850c6"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
```

Список объектов неполон по причинам, описанным выше

Практическое задание 8.1.5:

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

```
learning> 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
  },
```

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Практическое задание 8.1.6:

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

```
learning> db.unicorns.find({weight: {$gte: 500, $lte: 700}, gender: 'f'}, {_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 8.1.7:

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

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

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ `vampires`.

```
learning> db.unicorns.find({valmpires: {$exists: false}})
[
  {
    _id: ObjectId("6475c653baff4feb292850bd"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("6475c654baff4feb292850be"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("6475c654baff4feb292850bf"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
```

Практическое задание 8.1.9:

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

```
learning> db.unicorns.find({gender: 'm'}, {_id: 0, name: 1, loves: {$slice: 1}}).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' ] }
]
```

ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ

Практическое задание 8.2.1:

1) *Создайте коллекцию towns*

```
learning> arr = [{
...     name: 'Punxsutawney ',
...     populatiuon: 6200,
...     last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
...     famous_for: [ '' ],
...     mayor: { name: 'Jim Wehrle' }
...   },
...   {
...     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' }
...   },
... {
...     name: 'Portland',
...     populatiuon: 528000,
...     last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
...     famous_for: [ 'beer', 'food' ],
...     mayor: { name: 'Sam Adams', party: 'D' }
...   }
... ]
[
  {
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    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' }
  },
  {
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learning> db.towns.insert(arr)
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6475cf7cbaff4feb292850cc"),
    '1': ObjectId("6475cf7cbaff4feb292850cd"),
    '2': ObjectId("6475cf7cbaff4feb292850ce")
  }
}
```

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

```
learning> 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.*

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

КУРСОРЫ

Практическое задание 8.2.2:

1) *Сформировать функцию для вывода списка самцов единорогов.*
2) *Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.*

```
learning> let cursor = db.unicorns.find({gender: 'm'}).sort({name: 1}).limit(2); null;
null
```

3) *Вывести результат, используя forEach.*

```
learning> cursor.forEach(o => print(o))
{
  _id: ObjectId("6475c6aabaff4feb292850c8"),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId("6475c653baff4feb292850bd"),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

АГРЕГИРОВАННЫЕ ЗАПРОСЫ

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 8.2.4:

Вывести список предпочтений.

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

Практическое задание 8.2.5:

Посчитать количество особей единорогов обоих полов.

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

РЕДАКТИРОВАНИЕ ДАННЫХ

Практическое задание 8.2.6:

1) *Выполнить команду*

2) *Проверить содержимое коллекции* `unicorns`.

Практическое задание 8.2.7:

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

```
learning> db.unicorns.update({gender: 'f', name: 'Ayna'}, {$set: {weight: 800, vampires: 51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2. *Проверить содержимое коллекции* `unicorns`.

```
learning> db.unicorns.find({name: 'Ayna'})
[
  {
    _id: ObjectId("6475d564baff4feb292850d3"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

Практическое задание 8.2.8:

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

```
learning> db.unicorns.update({gender: 'm', name: 'Raleigh'}, {$addToSet: {loves: "redbull"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2. *Проверить содержимое коллекции* `unicorns`.

```
learning> db.unicorns.find({name: 'Raleigh'})
[
  {
    _id: ObjectId("6475d564baff4feb292850d5"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Практическое задание 8.2.9:

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

```
learning> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
```

2. *Проверить содержимое коллекции* `unicorns`.

```
learning> db.unicorns.find({gender: 'm'})
[
  {
    _id: ObjectId("6475d564baff4feb292850cf"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 73
  },
  {
    _id: ObjectId("6475d564baff4feb292850d1"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId("6475d564baff4feb292850d4"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 44
  },
```

Практическое задание 8.2.10:

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

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

2. *Проверить содержимое коллекции* `towns`.

```
learning> db.towns.find({})
[
  {
    _id: ObjectId("6475cf7cbaff4feb292850cc"),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId("6475cf7cbaff4feb292850cd"),
    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("6475cf7cbaff4feb292850ce"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

Практическое задание 8.2.11:

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

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

2. *Проверить содержимое коллекции* `unicorns`.

```
learning> db.unicorns.find({name: 'Pilot'})
[
  {
    _id: ObjectId("6475d565baff4feb292850d7"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

Практическое задание 8.2.12:

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

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

2. *Проверить содержимое коллекции* `unicorns`.

```
learning> db.unicorns.find({name: 'Aurora'})
[
  {
    _id: ObjectId("6475db2ca2a3541914043b54"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 8.2.13:

1) Создайте коллекцию towns, включающую следующие документы:

```
learning> towns = [{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"}}
... ]
[
  {
    name: 'Punxsutawney ',
    popujatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ 'phil the groundhog' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate("2009-07-31T00:00:00.000Z"),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learning> db.towns.insertMany(towns)
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6475dbe6a2a3541914043b55"),
    '1': ObjectId("6475dbe6a2a3541914043b56"),
    '2': ObjectId("6475dbe6a2a3541914043b57")
  }
}
```

2) *Удалите документы с беспартийными мэрами.*

```
learning> db.towns.deleteMany({"mayor.party": {$exists: 0}})
{ acknowledged: true, deletedCount: 1 }
```

3) *Проверьте содержание коллекции.*

```
learning> db.towns.find({})
[
  {
    _id: ObjectId("6475dbe6a2a3541914043b56"),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate("2009-07-31T00:00:00.000Z"),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId("6475dbe6a2a3541914043b57"),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

4) *Очистите коллекцию.*

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

5) *Просмотрите список доступных коллекций.*

```
learning> show collections
towns
unicorns
```

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.*

```
[
  {
    _id: 'US',
    name: 'United States of America',
    description: 'Пендосы...'
  },
  {
    _id: 'RU',
    name: 'Russia',
    description: 'Здесь живут древние Русы'
  },
  {
    _id: 'LZ',
    name: 'Lizards',
    description: 'Древние ящеры. Чешуйчатые ироды'
  }
]
learning> db.regions.insert(regions)
{
  acknowledged: true,
  insertedIds: { '0': 'US', '1': 'RU', '2': 'LZ' }
}
```

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

```
learning> db.unicorns.update({name: 'Dunx'}, {$set: {regions: {$ref: "regions", $id: "US"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learning> db.unicorns.update({name: 'Aurora'}, {$set: {regions: {$ref: "regions", $id: "RU"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learning> db.unicorns.update({name: 'Pilot'}, {$set: {regions: {$ref: "regions", $id: "LZ"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

3) Проверьте содержание коллекции единорогов.

```
[
  {
    _id: ObjectId("6475d565baff4feb292850d6"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId("6475d565baff4feb292850d7"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59,
    regions: DBRef("regions", 'LZ')
  },
  {
    _id: ObjectId("6475d565baff4feb292850d8"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("6475d565baff4feb292850d9"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 170,
    regions: DBRef("regions", 'US')
  },
  {
    _id: ObjectId("6475d5c3baff4feb292850da"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104
  },
  {
    _id: ObjectId("6475db2ca2a3541914043b54"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    regions: DBRef("regions", 'RU')
  }
]
```

НАСТРОЙКА ИНДЕКСОВ

Практическое задание 8.3.2:

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

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

Задать такой индекс можно.

УПРАВЛЕНИЕ ИНДЕКСАМИ

Практическое задание 8.3.3:

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

```
learning> 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.*

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

3) *Попытайтесь удалить индекс для идентификатора.*

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

Удаление индекса для идентификатора невозможно

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.*

```
learning> db.numbers.find({}).skip(99996)
[
  { _id: ObjectId("6475e4dca2a3541914060f3a"), value: 99996 },
  { _id: ObjectId("6475e4dca2a3541914060f3b"), value: 99997 },
  { _id: ObjectId("6475e4dca2a3541914060f3c"), value: 99998 },
  { _id: ObjectId("6475e4dca2a3541914060f3d"), value: 99999 }
]
```

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

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 39,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
```

4) *Создайте индекс для ключа* `value`.

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

5) *Получите информацию о всех индексах коллекции* `numbres`.

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

6) *Выполните запрос 2.*

```
learning> db.numbers.explain("executionStats").find({}).skip(99996)
```

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

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 37,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
```

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

Время запроса с индексом уменьшилось незначительно (конкретно – 2 мс), однако повторные запросы показывали различные результаты, в том числе и более долгие, чем изначальный запрос без индекса. Из это информации можно сделать вывод, что в данном случае индекс не оказывает существенного влияния на производительность.

Вывод:

В ходе лабораторной работы были изучены основы работы с СУБД MongoDB, в частности – настройка окружения (запуск Docker контейнера с СУБД), аутентификация, создание баз данных и коллекций, базовые CRUD операции. Также были был получен опыт работы с индексами в NoSQL базе данных.