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 «**Базы данных**»

Автор: Ким Даниил Дмитриевич

Факультет: ИКТ

Группа: K32391

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4 6.0.6

Практическое задание и выполнение:

Практическое задание 8.1.1:

1) Создайте базу данных learn.

```
test> show dbs
admin   40.00 KiB
config  12.00 KiB
local   40.00 KiB
test> use learn
switched to db learn
learn> db.createCollection('unicorns')
{ ok: 1 }
learn> db.unicorn.find().sort()

learn> db.unicorn.find().count()
(node:15201) [MONGODB DRIVER] Warning: cursor.count is deprecated and will be removed in the next major version, please use `collection.estimatedDocumentCount` or `collection.countDocuments` instead
(Use `node --trace-warnings ...` to show where the warning was created)
0
learn> 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("647cac770d65193480d67099") }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647cac770d65193480d6709a") }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647cac770d65193480d6709b") }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647cac770d65193480d6709c") }
}
learn> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647cac770d65193480d6709d") }
}
learn> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647cac770d65193480d6709e") }
}
learn> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647cac770d65193480d6709f") }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647cac790d65193480d670a0") }
}
```

2) Заполните коллекцию единорогов unicorns:

```
MongoServerError: Collection learn.unicorns already exists.
learn> db.unicorns.find().count()
12
learn> db.unicorns.find()
[
  {
    _id: ObjectId("647cac770d65193480d67099"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("647cac770d65193480d6709a"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("647cac770d65193480d6709b"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("647cac770d65193480d6709c"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("647cac770d65193480d6709d"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("647cac770d65193480d6709e"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
```

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

```
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>
```

4) Проверьте содержимое коллекции с помощью метода find()

```
  },
  {
    _id: ObjectId("647cb0d10d65193480d670a5"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
learn>
```

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.

```
learn> db.unicorns.find({gender:'f'}).sort({name:1}).limit(3)
[
  {
    _id: ObjectId("647cac770d65193480d6709a"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("647cac770d65193480d6709e"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId("647cac790d65193480d670a1"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn>
```

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

```
learn> db.unicorns.find({gender:'f', loves:'carrot'}, {_id:0, name:1}).sort({name:1}).limit(1)
[ { name: 'Aurora' } ]
learn>
```

```
[ { name: 'Aurora' } ]
learn> db.unicorns.find({gender:'f', loves:'carrot'})
[
  {
    _id: ObjectId("647cac770d65193480d6709a"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("647cac770d65193480d6709d"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("647cac790d65193480d670a3"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn>
```

```
]
learn> db.unicorns.find({gender:'f', loves:'carrot'}, {_id:0, name:1}).sort({name:1})
[ { name: 'Aurora' }, { name: 'Nimue' }, { name: 'Solnara' } ]
learn>
```

Практическое задание 8.1.3:

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

```
[ { name: 'Aurora' }, { name: 'Nimue' }, { name: 'Solnara' } ]
learn> db.unicorns.find({gender:'m'}, {loves:0, gender:0})
[
  {
    _id: ObjectId("647cac770d65193480d67099"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId("647cac770d65193480d6709b"),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId("647cac770d65193480d6709c"),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId("647cac770d65193480d6709f"),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId("647cac790d65193480d670a0"),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId("647cac790d65193480d670a2"),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId("647cad440d65193480d670a4"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId("647cb0d10d65193480d670a5"),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
]
learn>
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find({}).sort({$natural: -1})
[
  {
    _id: ObjectId("647cb0d10d65193480d670a5"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("647cad440d65193480d670a4"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("647cac790d65193480d670a3"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("647cac790d65193480d670a2"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId("647cac790d65193480d670a1"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId("647cac790d65193480d670a0"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
```

Практическое задание 8.1.5:

Вывести список единорогов с названием первого любимого предпочтения,

исключив идентификатор.

```
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.

```
learn> db.unicorns.find({gender:'f', weight:{$gt: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'
  }
]
learn>
```

Практическое задание 8.1.7:

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

```
]
learn> db.unicorns.find({gender:'m', weight:{$gt: 500}, loves:{$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.

```
learn> db.unicorns.find({gender:'f', vampires:{$exists:false}}, {_id:0, name:1})
[ { name: 'Nimue' } ]
learn>
```

Практическое задание 8.1.9:

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

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

Практическое задание 8.2.1:

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.insert({name: "Punxsutawney ", populatiuon: 6200, last_sensus: ISODate("2008-01-31"), famous_for: [""], mayor: {name: "Jim Wehrle" }})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647cd8660d65193480d670a6") }
}
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("647cd8a90d65193480d670a7") }
}
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("647cd8bd0d65193480d670a8") }
}
learn> db.towns.find().count()
3
learn>
```

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}).sort({name:1}
)
[
  {
    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.

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

<u>Практическое задание 8.2.2:</u>

3) Сформировать функцию для вывода списка самцов единорогов.

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

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

```
learn> males = function() {return this.gender == 'm';}
[Function: males]
learn> var cursor = db.unicorns.find({'$where':males}).sort({name:1}).limit(2);
null
```

5) Вывести результат, используя forEach.

```
learn> cursor.forEach(function(obj) {print(obj)})
{
  _id: ObjectId("647cb0d10d65193480d670a5"),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId("647cad440d65193480d670a4"),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}

learn>
```

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 8.2.4:

Вывести список предпочтений.

```
2
learn> db.unicorns.distinct('loves')
[
  'apple',      'carrot',
  'chocolate',  'energon',
  'grape',      'lemon',
  'papaya',     'redbull',
  'strawberry', 'sugar',
  'watermelon'
]
learn>
```

Практическое задание 8.2.5:

Подсчитать количество особей единорогов обоих полов.

```
learn> db.unicorns.aggregate({'$group':{_id:"$gender", count:{$sum:1}}})
[ { _id: 'm', count: 8 }, { _id: 'f', count: 5 } ]
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.

```
learn> db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
TypeError: db.unicorns.save is not a function
learn> db.unicorns.insert({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})

{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647cdf030d65193480d670a9") }
}
learn> db.unicorns.find().count()
14
learn>
```

Практическое задание 8.2.7:

1. Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51
вампира.
2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.updateOne({name:'Ayna'}, {$set: {weight:800, vampires:51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:'Ayna'})
[
  {
    _id: ObjectId("647cac770d65193480d6709e"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
learn>
```

Практическое задание 8.2.8:

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

```
learn> db.unicorns.updateOne({name:'Raleigh'}, {$set:{loves:['redbull']}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:'Raleigh'})
[
  {
    _id: ObjectId("647cac790d65193480d670a0"),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
learn>
```

Практическое задание 8.2.9:

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.
2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.find({gender:'m'})
[
  {
    _id: ObjectId("647cac770d65193480d67099"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("647cac770d65193480d6709b"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("647cac770d65193480d6709c"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("647cac770d65193480d6709f"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId("647cac790d65193480d670a0"),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
```

```
learn> db.unicorns.updateMany({gender:'m'}, {$inc:{vampire:5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 9,
  modifiedCount: 9,
  upsertedCount: 0
}
learn> db.unicorns.find({gender:'m'}, {_id:0, name:1, vampires:1}).sort({name:1})
[
  { name: 'Barny' },
  { name: 'Dunx', vampires: 165 },
  { name: 'Horny', vampires: 63 },
  { name: 'Horny', vampires: 63 },
  { name: 'Kenny', vampires: 39 },
  { name: 'Pilot', vampires: 54 },
  { name: 'Raleigh', vampires: 2 },
  { name: 'Roooooodles', vampires: 99 },
  { name: 'Unicrom', vampires: 182 }
]
learn>
```

Практическое задание 8.2.10:

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

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

learn> db.towns.find({name:'Portland'})
[
  {
    _id: ObjectId("647cd8bd0d65193480d670a8"),
    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.
2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.updateOne({name:'Pilot'}, {$push:{loves:'chocolate'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:'Pilot'})
[
  {
    _id: ObjectId("647cac790d65193480d670a2"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 54,
    vampire: 5
  }
]
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.

```
learn> db.unicorns.updateOne({name:'Aurora'}, {$addToSet:{loves: {$each:['sugar', 'lem
on']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:'Aurora})
Uncaught:
SyntaxError: Unterminated string constant. (1:23)

> 1 | db.unicorns.find({name:'Aurora})
    |                       ^
  2 |

learn> db.unicorns.find({name:'Aurora'})
[
  {
    _id: ObjectId("647cac770d65193480d6709a"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

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) Проверьте содержание коллекции.
4) Очистите коллекцию.
5) Просмотрите список доступных коллекц

```
learn> db.towns.find().count()
3
learn> db.towns.find()
[
  {
    _id: ObjectId("647cd8660d65193480d670a6"),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId("647cd8a90d65193480d670a7"),
    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("647cd8bd0d65193480d670a8"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
learn> db.towns.deleteMany({'mayor.party': {$exists:false}})
{ acknowledged: true, deletedCount: 2 }
learn> db.towns.find().count()
1
learn> db.towns.drop()
true
learn> show collections
unicorns
learn>
```

Практическое задание 8.3.1:

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

```
learn> db.areas.insertMany([{'_id':'Rainbow', 'name':'Cute Rainbow', 'description':'So
 cuuute'}, {_id:'Garden', 'name':'Red garden', 'description':'Far from here'}])
{ acknowledged: true, insertedIds: { '0': 'Rainbow', '1': 'Garden' } }
learn> db.areas.find()
[
  { _id: 'Rainbow', name: 'Cute Rainbow', description: 'So cuuute' },
  { _id: 'Garden', name: 'Red garden', description: 'Far from here' }
]
learn>
```

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

```
learn> db.unicorns.updateOne({name:'Horny'}, {$set:{area:{$ref:'areas', $id:'Rainbow'}
}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:'Horny'})
[
  {
    _id: ObjectId("647cac770d65193480d67099"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63,
    vampire: 5,
    area: DBRef("areas", 'Rainbow')
  },
  {
    _id: ObjectId("647cad440d65193480d670a4"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63,
    vampire: 5
  }
]
learn>
```

3) Проверьте содержание коллекции единорогов

```
learn> db.unicorns.find({area: {$exists:true}}, {_id:0, name:1, area:1})
[ { name: 'Horny', area: DBRef("areas", 'Rainbow') } ]
learn>
```

Практическое задание 8.3.2:

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' ]
learn>
```

Практическое задание 8.3.3:

1) Получите информацию о всех индексах коллекции unicorns .

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

2) Попытайтесь удалить индекс для идентификатора.

```
learn> db.unicorns.dropIndex('name_1')
{ nIndexesWas: 2, ok: 1 }
learn> db.unicorns.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn> db.unicorns.dropIndexes('_id')
MongoshInternalError: index not found with name [_id]
learn> db.unicorns.dropIndexes('_id_')
MongoServerError: cannot drop _id index
learn>
```

Практическое задание 8.3.4:

1) Создайте объемную коллекцию numbers, задействовав курсор:

```
learn> db.createCollection('numbers')
{ ok: 1 }
learn> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
```

```
learn> db.numbers.find()
[
  { _id: ObjectId("647cf6110d65193480d670aa"), value: 0 },
  { _id: ObjectId("647cf6110d65193480d670ab"), value: 1 },
  { _id: ObjectId("647cf6110d65193480d670ac"), value: 2 },
  { _id: ObjectId("647cf6110d65193480d670ad"), value: 3 },
  { _id: ObjectId("647cf6110d65193480d670ae"), value: 4 },
  { _id: ObjectId("647cf6110d65193480d670af"), value: 5 },
  { _id: ObjectId("647cf6110d65193480d670b0"), value: 6 },
  { _id: ObjectId("647cf6110d65193480d670b1"), value: 7 },
  { _id: ObjectId("647cf6110d65193480d670b2"), value: 8 },
  { _id: ObjectId("647cf6110d65193480d670b3"), value: 9 },
  { _id: ObjectId("647cf6110d65193480d670b4"), value: 10 },
  { _id: ObjectId("647cf6110d65193480d670b5"), value: 11 },
  { _id: ObjectId("647cf6110d65193480d670b6"), value: 12 },
  { _id: ObjectId("647cf6110d65193480d670b7"), value: 13 },
  { _id: ObjectId("647cf6110d65193480d670b8"), value: 14 },
  { _id: ObjectId("647cf6110d65193480d670b9"), value: 15 },
  { _id: ObjectId("647cf6110d65193480d670ba"), value: 16 },
  { _id: ObjectId("647cf6110d65193480d670bb"), value: 17 },
  { _id: ObjectId("647cf6110d65193480d670bc"), value: 18 },
  { _id: ObjectId("647cf6110d65193480d670bd"), value: 19 }
]
Type "it" for more
learn> it
[
  { _id: ObjectId("647cf6110d65193480d670be"), value: 20 },
  { _id: ObjectId("647cf6110d65193480d670bf"), value: 21 },
  { _id: ObjectId("647cf6110d65193480d670c0"), value: 22 },
  { _id: ObjectId("647cf6110d65193480d670c1"), value: 23 },
  { _id: ObjectId("647cf6110d65193480d670c2"), value: 24 },
  { _id: ObjectId("647cf6110d65193480d670c3"), value: 25 },
  { _id: ObjectId("647cf6110d65193480d670c4"), value: 26 },
  { _id: ObjectId("647cf6110d65193480d670c5"), value: 27 },
  { _id: ObjectId("647cf6110d65193480d670c6"), value: 28 },
  { _id: ObjectId("647cf6110d65193480d670c7"), value: 29 },
  { _id: ObjectId("647cf6110d65193480d670c8"), value: 30 },
  { _id: ObjectId("647cf6110d65193480d670c9"), value: 31 },
  { _id: ObjectId("647cf6110d65193480d670ca"), value: 32 },
  { _id: ObjectId("647cf6110d65193480d670cb"), value: 33 },
  { _id: ObjectId("647cf6110d65193480d670cc"), value: 34 },
  { _id: ObjectId("647cf6110d65193480d670cd"), value: 35 },
  { _id: ObjectId("647cf6110d65193480d670ce"), value: 36 },
  { _id: ObjectId("647cf6110d65193480d670cf"), value: 37 },
  { _id: ObjectId("647cf6110d65193480d670d0"), value: 38 },
  { _id: ObjectId("647cf6110d65193480d670d1"), value: 39 }
]
Type "it" for more
learn>
```

2) Выберите последних четыре документа.
3) Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра executionTimeMillis)
4) Создайте индекс для ключа value.
5) Получите информацию о всех индексах коллекции numbres.
6) Выполните запрос 2.
7) Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?
8) Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

```
learn> db.numbers.explain("executionStats").find().sort({$natural:-1}).limit(4)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: { stage: 'COLLSCAN', direction: 'backward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
```

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

```
learn> db.numbers.find().sort({_id:-1}).limit(4)
[
  { _id: ObjectId("647cf63a0d65193480d7f749"), value: 99999 },
  { _id: ObjectId("647cf63a0d65193480d7f748"), value: 99998 },
  { _id: ObjectId("647cf63a0d65193480d7f747"), value: 99997 },
  { _id: ObjectId("647cf63a0d65193480d7f746"), value: 99996 }
]
learn>
```

```
learn> db.numbers.explain("executionStats").find().sort({$natural:-1}).limit(4)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
```

```
> db.numbers.find().sort({_id: -1}).limit(4)
db.numbers.find().sort({_id: -1}).limit(4)
```

```
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
```

С индексом запрос работает быстрее.

Выводы:

В процессе работы были получены навыки работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.