

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

Выполнил студент:

Зайцев Кирилл Дмитриевич

Группа №К3 3402

Преподаватель:

Говорова Марина Михайловна

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.

Программное обеспечение: СУБД MongoDB 4+, 6.0.6 (текущая).

Ход работы:

Практическое задание 8.1.1

1. Создайте базу данных learn.

```
>_MONGOSH
> use learn
< switched to db learn
learn>|
```

use learn

2. Заполните коллекцию единорогов unicorns:

```
>_MONGOSH
> use learn
< switched to db 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: 'Unicorn', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182));
db.unicorns.insert((name: 'Roooooodies', 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("64768e4e193ea035d021dfd7")
  }
}
learn>|
>_MONGOSH
> db.unicorns.insertOne((name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165))
< {
  acknowledged: true,
  insertedId: ObjectId("64768e96193ea035d021dfd8")
}
learn>|
```

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

3. Проверьте содержимое коллекции с помощью метода find.

```

>_MONGOOSH
> db.unicorns.findOne({name: 'Dunx'})
< {
  _id: ObjectId("64760e96103ea035d021dfds"),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn>

```

```
{name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
```

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.

```

>_MONGOOSH
{
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId("64760e4e103ea035d021dfcd"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId("64760e4e103ea035d021dfd3"),
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
learn>

```

```
db.unicorns.find({gender: 'm'}).limit(3).sort({name: 1})
```

```

>_MONGOOSH
> db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
< {
  _id: ObjectId("64760e4e103ea035d021dfce"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId("64760e4e103ea035d021dfd2"),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 733,
  gender: 'f',
  vampires: 40
}
{
  _id: ObjectId("64760e4e103ea035d021dfd5"),
  name: 'Leia',
  loves: [

```

```
db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
```

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

```
>_MONGOOSH
> db.unicorns.findOne({loves: "carrot"})
< {
  _id: ObjectId("64760e4e103ea035d021dfcd"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

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

Практическое задание 8.1.3

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

```
>_MONGOOSH
> db.unicorns.find({gender: 'm'}, {loves: 0, _id: 0})
< [
  {
    name: 'Horny',
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Unicrom',
    weight: 984,
    gender: 'm',
    vampires: 102
  },
  {
    name: 'Rooooooodles',
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    name: 'Kenny',
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    name: 'Raleigh',

```

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

Практическое задание 8.1.4

Вывести список единорогов в обратном порядке добавления.

```
>_MONGOOSH
> db.unicorns.find().sort({$natural: -1})
< [
  {
    _id: ObjectId("64760e4e103ea035d021dfd8"),
    name: 'Dunk',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("64760e4e103ea035d021dfd7"),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("64760e4e103ea035d021dfd6"),
    name: 'Pilot',
    loves: [
      'apple',

```

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

Практическое задание 8.1.5

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

```
>_MONGOSH
> db.unicorns.find({}, {_id: 0, loves: {$slice: 1}})
< {
  name: 'Horny',
  loves: [
    'carrot'
  ],
  weight: 690,
  gender: 'm',
  vampires: 63
}
{
  name: 'Aurora',
  loves: [
    'carrot'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  name: 'Unicom',
  loves: [
    'energon'
  ],
  weight: 984,
  gender: 'm',
  vampires: 182
}
```

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

Практическое задание 8.1.6

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

```
>_MONGOSH
> 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: 601,
  gender: 'f',
  vampires: 33
}
```

```
db.unicorns.find({gender: 'f', weight: {$gt: 500, $lt: 700}}, {_id: 0})
```

Практическое задание 8.1.7

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

```

>_MONGOOSH
>
> 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>|

```

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

Практическое задание 8.1.8

Найти всех единорогов, не имеющих ключ vampires.

```

>_MONGOOSH
> db.unicorns.find({vampires: {$exists: false}})
< {
  _id: ObjectId("64760e4e103ee035d021dfd7"),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
learn>

```

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

Практическое задание 8.1.9

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

```

>_MONGOOSH
> db.unicorns.find((gender: 'm'), {_id: 0, name: 1, loves: {$slice: 1}}).sort({name: 1})
< {
  name: 'Dunk',
  loves: [
    'grape'
  ]
}
{
  name: 'Horny',
  loves: [
    'carrot'
  ]
}
{
  name: 'Kenny',
  loves: [
    'grape'
  ]
}
{
  name: 'Pilot',
  loves: [
    'apple'
  ]
}
{
  name: 'Raleigh',

```

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

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"}}

```

```

>_MONGOSH
mayor: {
  name: "Jim Wehrle"
}},
{name: "New York",
population: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
  name: "Michael Bloomberg",
  party: "I"}},
{name: "Portland",
population: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
  name: "Sam Adams",
  party: "D"}}
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("647615cc103ea035d021dfd9"),
    '1': ObjectId("647615cc103ea035d021dfd9"),
    '2': ObjectId("647615cc103ea035d021dfd9")
  }
}
learn>

```

```

db.towns.insertMany([ {name:
"Punxsutawney ",
population: 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"}}
])

```

- 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"}, {_id: 0, name: 1, mayor: 1})
< {
  name: 'New York',
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
```

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

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

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

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

Практическое задание 8.2.2

- Сформировать функцию для вывода списка самцов единорогов.

```
> MONGOSH
> var logUnicorns = function(cursor) {}
> let logUnicorns = function(cursor) {
  cursor.forEach((el) => {
    print(el.name)
  })
};

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

logUnicorns(cursor)
< Dunk
< Horny
learn>
```

```
let logUnicorns = function(cursor) {
  cursor.forEach((el) => {
    print(el.name)
  })
};

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

logUnicorns(cursor)
```

- Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.
- Вывести результат, используя forEach.

Практическое задание 8.2.3

Вывести количество самок единорогов весом от полутонны до 600 кг.


```

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

```

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

Практическое задание 8.2.4

Вывести список предпочтений.

```

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

```

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

Практическое задание 8.2.5

Посчитать количество особей единорогов обоих полов.

```

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

```

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

Практическое задание 8.2.6

1. Выполнить команду:

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

Проверить содержимое коллекции unicorns.

```

>_MONGOSH
> db.unicorns.find().count()
< 12
> db.unicorns.replaceOne({name: 'Barney', loves: ['grape'],
weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
> db.unicorns.find().count()
< 12
learn>

```

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

Практическое задание 8.2.7

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

```
> db.unicorns.updateOne({name: 'Ayna'}, {$set: {weight: 800, vampires: 51}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.findOne({name: 'Ayna'})
< {
  _id: ObjectId("64760e4e103ea035d021dfd2"),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 800,
  gender: 'f',
  vampires: 51
}
```

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

Практическое задание 8.2.8

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

```
> MONGODB
> db.unicorns.updateOne({name: 'Raleigh', gender: 'm'}, {$push: {loves: 'redbull'}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.findOne({name: 'Raleigh', gender: 'm'})
< {
  _id: ObjectId("64760e4e103ea035d021dfd4"),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar',
    'redbull'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
```

```
db.unicorns.updateOne({name: 'Raleigh', gender: 'm'}, {$push: {loves: 'redbull'}})
```

Практическое задание 8.2.9

Всем самцам единорогов увеличить количество убитых вампиров на 5.

```

>_MONGOOSH
> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
> db.unicorns.find({gender: 'm'})
< {
  _id: ObjectId("64760e4e103ea035d021dfcd"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 68
}
{
  _id: ObjectId("64760e4e103ea035d021dfcf"),
  name: 'Umicrom',
  loves: [
    'energon',
    'redbull'
  ],
  weight: 984,
  gender: 'm',
  vampires: 53
}

```

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

Практическое задание 8.2.11

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

```

>_MONGOOSH
> db.unicorns.updateOne({name: 'Pilot', gender: 'm'}, {$push: {loves: 'Chocolate'}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.find({name: 'Pilot', gender: 'm'})
< {
  _id: ObjectId("64760e4e103ea035d021dfd6"),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'Chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
learn>

```

```
db.unicorns.updateOne({name: 'Pilot', gender: 'm'}, {$push: {loves: 'Chocolate'}})
```

Практическое задание 8.1.12

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

```

>_MONGOOSH
> db.unicorns.updateOne({name: 'Aurora', gender: 'f'}, {$addToSet: {loves: {$each: ['sugar', 'lemons']}}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.find({name: 'Aurora', gender: 'f'})
< {
  _id: ObjectId("64760e4e103ea035d021dfce"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape',
    'sugar',
    'lemons'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>

```

```
db.unicorns.updateOne({name: 'Aurora', gender:'f'}, {$addToSet: {loves: {$each: ['sugar', 'lemons']}}})
```

Практическое задание 8.1.13

1. Удалите документы с беспартийными мэрами.

```
>_MONGOOSH
> db.towns.deleteMany({"mayor.party": {$exists: false}})
< {
  acknowledged: true,
  deletedCount: 1
}
> db.towns.find({})
< {
  _id: ObjectId("64772c310997c99920fab103"),
  name: 'New York',
  population: 22200000,
  last_census: 2009-07-31T00:00:00.000Z,
  famous_for: [
    'status of liberty',
    'food'
  ],
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
{
  _id: ObjectId("64772c310997c99920fab104"),
  name: 'Portland',
  population: 520000,
  last_census: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Tom Adams',
    party: 'I'
  }
}
```

- 2.
3. `db.towns.deleteMany({"mayor.party": {$exists: false}})`

4. Проверьте содержание коллекции.
5. Очистите коллекцию.

```
>_MONGOOSH
> db.towns.deleteMany({})
< {
  acknowledged: true,
  deletedCount: 2
}
> show collections
< towns
unicorns
learn>

db.towns.deleteMany({})
```

6. Просмотрите список доступных коллекций.

Практическое задание 8.3.1

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

```
>_MONGOOSH
> db.unicorn_area.insertMany([
  { _id: 'areal_field', name: 'Field', description: 'description.....' },
  { _id: 'areal_forest', name: 'Forest', description: 'description.....' },
  { _id: 'areal_lake', name: 'Lake', description: 'description.....' },
])
< {
  acknowledged: true,
  insertedIds: {
    '0': 'areal_field',
    '1': 'areal_forest',
    '2': 'areal_lake'
  }
}
learn>
```

```
db.unicorn_area.insertMany([
  { _id: 'areal_field', name: 'Field', description: 'description.....' },
```

```
{_id: 'areal_forest', name: 'Forest', description: 'description.....'},
{_id: 'areal_lake', name: 'Lake', description: 'description.....'}, ])
```

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

```
>_MONGOSH
> db.unicorns.update({name: 'Pilot'}, {$set: {area: {$ref: 'unicorn_area', $id: 'area_lake'}}})
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.find({name: 'Pilot'})
< {
  _id: ObjectId("64760e4e103ea035d921dfds"),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'Chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59,
  area: DBRef("unicorn_area", 'area_lake')
}
learn>
```

```
db.unicorns.update({name: 'Pilot'}, {$set: {area: {$ref: 'unicorn_area', $id: 'area_lake'}}})
```

- Проверьте содержание коллекции единорогов.

Практическое задание 8.3.2

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

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

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

Практическое задание 8.3.3

- Получите информацию о всех индексах коллекции unicorns.
- Удалите все индексы, кроме индекса для идентификатора.
- Попытайтесь удалить индекс для идентификатора.

```
>_MONGOSH
> db.unicorns.getIndexes()
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
> db.unicorns.dropIndex('name_1')
< { nIndexesWas: 2, ok: 1 }
> db.unicorns.getIndexes()
< [ { v: 2, key: { _id: 1 }, name: '_id_' } ]
> db.unicorns.dropIndex('_id_')
MongoServerError: cannot drop _id index
learn>
```

```
db.unicorns.getIndexes() db.unicorns.dropIndex('name_1')
```

Практическое задание 8.3.4

1. Создайте объемную коллекцию numbers, задействовав курсор:

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

```
>_MONGOOSH
> for(let i = 0; i < 100000; i++){db.numbers.insert({value: i})}
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6477313a0997c99920fc37a4")
  }
}
> db.numbers.find().count()
< 100000
learn>
```

2. Выберите последних четыре документа.
3. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра `executionTimeMillis`) Запрос занял 0мс

```
>_MONGOOSH
> db.numbers.ensureIndex({name: 1})
< [ 'name_1' ]
> db.numbers.getIndexes
< [Function: getIndexes] AsyncFunction {
  apiVersions: [ 1, Infinity ],
  serverVersions: [ '3.2.0', '999.999.999' ],
  returnsPromise: true,
  topologies: [ 'RepSet', 'Sharded', 'LoadBalanced', 'Standalone' ],
  returnType: { type: 'unknown', attributes: {} },
  deprecated: false,
  platforms: [ 'Compass', 'Browser', 'CLI' ],
  isDirectShellCommand: false,
  acceptsRawInput: false,
  shellCommandCompleter: undefined,
  help: [Function (anonymous)] Help
}
> db.numbers.getIndexes()
MongoServerError: ns does not exist: learn.numbers
> db.numbers.getIndexes()
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1' }
]
learn>
```

```
db.numbers.explain('executionStats').find({}).sort({$natural: -1}).limit(4)
db.numbers.ensureIndex({name: 1}) db.numbers.getIndexes()
```

4. Создайте индекс для ключа value.
5. Получите информацию о всех индексах коллекции numbers.
6. Выполните запрос 2.

```
>_MONOOSH
> 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,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 0
  }
}
```

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

Запрос также занял 0 мс

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

Время выполнения запроса не изменилось. Возможно в этом случае индекс не ускоряет выполнение запроса.

Выводы:

1. Были произведены операции вставки, удаления, обновления, выборки данных.
2. Были созданы связи между объектами разных коллекций с помощью DBRef.
3. Были созданы индексы, сравнены запросы до и после применения индексов.