

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

Отчет

по Лабораторной Работе № 5

по дисциплине «Проектирование и реализация баз данных»

Автор: Тихонова Ксения Евгеньевна

Факультет: ФИКТ

Группа: К32402

Преподаватель: Говорова Марина Михайловна

itmo

Санкт-Петербург

2023

ПРАКТИЧЕСКОЕ ЗАДАНИЕ 8.1

Практическое задание 8.1.1:

Создаем базу данных learn

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

В ней создаем коллекцию unicorns

```
> db.createCollection("unicorns")  
< { ok: 1 }
```

Добавляем элемент коллекции

```
> 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("6494b6bf611cadd3df60cd72")  
  }  
}
```

Добавляем много элементов коллекции за раз

```

> db.unicorns.insertMany([
  {name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43},
  {name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182},
  {name: 'Rooodoodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99},
  {name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80},
  {name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40},
  {name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39},
  {name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2},
  {name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33},
  {name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54},
  {name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'},
])
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6494b8db611cadd3df60cd9d"),
    '1': ObjectId("6494b8db611cadd3df60cd9e"),
    '2': ObjectId("6494b8db611cadd3df60cd9f"),
    '3': ObjectId("6494b8db611cadd3df60cda0"),
    '4': ObjectId("6494b8db611cadd3df60cda1"),
    '5': ObjectId("6494b8db611cadd3df60cda2"),
    '6': ObjectId("6494b8db611cadd3df60cda3"),
    '7': ObjectId("6494b8db611cadd3df60cda4"),
    '8': ObjectId("6494b8db611cadd3df60cda5"),
    '9': ObjectId("6494b8db611cadd3df60cda6")
  }
}

```

Проверяем количество элементов

```

> db.unicorns.find().count()
< 11

```

Вторым способом добавляем ещё один элемент коллекции

```

> document={name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
< {
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
> db.unicorns.insert(document)
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6494b9bc611cadd3df60cda7")
  }
}

```

Проверяем количество ещё раз

```
> db.unicorns.countDocuments()  
< 12
```

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.

Самцы и самки единорогов:

```
> db.unicorns.find({gender: 'm'})  
< {  
  _id: ObjectId("6494b888611cadd3df60cd9c"),  
  name: 'Horny',  
  loves: [  
    'carrot',  
    'papaya'  
  ],  
  weight: 600,  
  gender: 'm',  
  vampires: 63  
}  
{  
  _id: ObjectId("6494b8db611cadd3df60cd9e"),  
  name: 'Unicrom',  
  loves: [  
    'energon',  
    'redbull'  
  ],  
  weight: 984,  
  gender: 'm',  
  vampires: 182  
}
```

```
> db.unicorns.find({gender: 'f'})
< {
  _id: ObjectId("6494b8db611cadd3df60cd9d"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId("6494b8db611cadd3df60cda0"),
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],
  weight: 550,
  gender: 'f',
  vampires: 80
}
{
  _id: ObjectId("6494b8db611cadd3df60cda1"),
  name: 'Ayna',
```

Ограничиваем список самок до трёх:

```
> db.unicorns.find({gender: 'f'}).limit(3)
< {
  _id: ObjectId("6494b8db611cadd3df60cd9d"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId("6494b8db611cadd3df60cda0"),
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],
  weight: 550,
  gender: 'f',
  vampires: 80
}
{
  _id: ObjectId("6494b8db611cadd3df60cda1"),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 600,
  gender: 'f',
  vampires: 100
}
```

Сортируем по имени:

```

> db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
< {
  _id: ObjectId("6494b8db611cadd3df60cd9d"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId("6494b8db611cadd3df60cda1"),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 733,
  gender: 'f',
  vampires: 40
}
{
  _id: ObjectId("6494b8db611cadd3df60cda4"),
  name: 'Leia',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 500,
  gender: 'f',
  vampires: 35
}

```

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

Нахождение через *limit*

```
> db.unicorns.find({loves: "carrot"}).limit(1)
< {
  _id: ObjectId("6494b888611cadd3df60cd9c"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

Нахождение через findOne

```
> db.unicorns.findOne({loves: "carrot"})
< {
  _id: ObjectId("6494b888611cadd3df60cd9c"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

Практическое задание 8.1.3:

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


```
> db.unicorns.find({gender: 'm'}, {likes: false, gender: false})
< {
  _id: ObjectId("6494b888611cadd3df60cd9c"),
  name: 'Horny',
  weight: 600,
  vampires: 63
}
{
  _id: ObjectId("6494b8db611cadd3df60cd9e"),
  name: 'Unicrom',
  weight: 984,
  vampires: 182
}
{
  _id: ObjectId("6494b8db611cadd3df60cd9f"),
  name: 'Rooooooodles',
  weight: 575,
  vampires: 99
}
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find().sort({$natural: -1}).limit(3);
< {
  _id: ObjectId("6494b9bc611cadd3df60cda7"),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId("6494b8db611cadd3df60cda6"),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  _id: ObjectId("6494b8db611cadd3df60cda5"),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 650,
```

Практическое задание 8.1.5:

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

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

Практическое задание 8.1.6:

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

```

> db.unicorns.find({weight: {$gte: 500, $lt: 700}}, {_id: false})
< {
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  name: 'Rooooooodles',
  loves: [
    'apple'
  ],
  weight: 575,
  gender: 'm',
  vampires: 99
}

```

Практическое задание 8.1.7:

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

```

> db.unicorns.find({gender: 'm', weight: {$gte: 500}, loves: {$all: ["grape", "lemon"]}})
< {
  _id: ObjectId("64953d68c6e023f3fe8b8c6f"),
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],

```

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ vampires.

```

> db.unicorns.find({vampires: {$exists:false}})
< {
  _id: ObjectId("6494b8db611cadd3df60cda6"),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}

```

Практическое задание 8.1.9:

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

```

> db.unicorns.find({gender: 'm', weight: {$gte: 500}, loves: {$all: ["grape", "lemon"]}})
< {
  _id: ObjectId("64953d68c6e023f3fe8b8c6f"),
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 44
}
> db.unicorns.find({gender: 'm'}, {loves: {$slice: 1}}).sort({name: 1})
< {
  _id: ObjectId("64953d93c6e023f3fe8b8c74"),
  name: 'Dunx',
  loves: [
    'grape'
  ],
  weight: 704,
  gender: 'm',
  vampires: 170
}
{
  _id: ObjectId("64953d68c6e023f3fe8b8c69"),
  name: 'Horny',
  loves: [
    'carrot'
  ],
  weight: 540,
  gender: 'f',
  vampires: 44
}

```

ПРАКТИЧЕСКОЕ ЗАДАНИЕ 2

Практическое задание 8.2.1:

1) Создайте коллекцию *towns*, включающую следующие документы:

```
> db.createCollection("towns")  
  
< { ok: 1 }
```

```
> 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("64952c4bfaee397ba77edda4"),  
    '1': ObjectId("64952c4bfaee397ba77edda5"),  
    '2': ObjectId("64952c4bfaee397ba77edda6")  
  }  
}
```

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

```
> db.towns.find({"mayor.party": "I"}, {_id: false, name: true, "mayor.name":true})
```

```
< {  
  name: 'New York',  
  mayor: {  
    name: 'Michael Bloomberg'  
  }  
}
```

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

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

```
< {  
  name: 'Punxsutawney ',  
  mayor: {  
    name: 'Jim Wehrle'  
  }  
}
```

Практическое задание 8.2.2:

- 1) Сформировать функцию для вывода списка самцов единорогов.

```
> fn = function(){  
  db.unicorns.find({gender:'m'}, {_id:0}).sort({name: 1 }).forEach((unicorn) =>  
    {print(unicorn.name);});}  
< [Function: fn]
```

Вывод:

```
> fn()  
< Dunx  
< Horny  
< Kenny  
< Pilot  
< Raleigh  
< Rooooooodles  
< Unicrom
```

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

```
> cursor = db.unicorns.find().sort({name: 1}).limit(2);null;  
< null
```

3) Вывести результат, используя `forEach`.

```
> cursor.forEach(unicorn => {print(unicorn.name);})  
< Aurora  
< Айна
```

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
> db.unicorns.find({weight: {$gte: 500, $lt: 600}}).count()  
< 3
```

Практическое задание 8.2.4:

Вывести список предпочтений.


```
> 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.

```
> db.unicorns.find({gender: {$in: ['m', 'f']}}).count()
< 12
```

Считаем отдельно по каждому полу:

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

Практическое задание 8.2.6:

1. Выполнить команду:

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

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

```
✖ ▶ TypeError: db.unicorns.save is not a function
```

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

```
✖ ▶ MongoInvalidArgumentError: Update document requires atomic operators
```

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

```
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
```

```
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64953b9c81b1f38a518691ab")
  }
}
```

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.count()
```

```
< DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
```

```
< 13
```

Практическое задание 8.2.7:

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

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

```
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2. Проверить содержимое коллекции unicorns.

```
{
  _id: ObjectId("64953d68c6e023f3fe8b8c6e"),
  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.

```
> db.unicorns.update({name: "Raleigh"}, {$set: {loves: ['redbull']}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2. Проверить содержимое коллекции unicorns.

```
{
  _id: ObjectId("64953d68c6e023f3fe8b8c70"),
  name: 'Raleigh',
  loves: [
    'redbull'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
```

Практическое задание 8.2.9:

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

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

2. Проверить содержимое коллекции *unicorns*.

```
{
  _id: ObjectId("64953d68c6e023f3fe8b8c70"),
  name: 'Raleigh',
  loves: [
    'redbull'
  ],
  weight: 421,
  gender: 'm',
  vampires: 7
}
{
  _id: ObjectId("64953d68c6e023f3fe8b8c71"),
  name: 'Leia',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 601,
  gender: 'f',
  vampires: 33
}
```

Практическое задание 8.2.10:

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

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

2. Проверить содержимое коллекции towns.

```
{
  _id: ObjectId("64953ff2c6e023f3fe8b8c77"),
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: 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.

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

2. Проверить содержимое коллекции unicorns.

```
{
  _id: ObjectId("64953d68c6e023f3fe8b8c72"),
  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.

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

2. Проверить содержимое коллекции unicorns.

```
{
  _id: ObjectId("64953d68c6e023f3fe8b8c6a"),
  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:*

```
{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.*

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

3. *Проверьте содержание коллекции.*


```

    name: 'New York',
    population: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [
      'status of liberty',
      'food'
    ],
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
}
{
  _id: ObjectId("6495410bc6e023f3fe8b8c7a"),
  name: 'Portland',
  population: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams',
    party: 'D'
  }
}

```

4. *Очистите коллекцию.*

```

> db.towns.remove({})

< {
  acknowledged: true,
  deletedCount: 2
}

```

5. *Просмотрите список доступных коллекций.*

```

> db.getCollectionNames()

< [ 'unicorns', 'towns' ]

```

ПРАКТИЧЕСКОЕ ЗАДАНИЕ 8.3

Практическое задание 8.3.1:

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

```

> db.createCollection("unicorn_habitat")
< { ok: 1 }
> db.unicorn_habitat.insertMany([
  {
    _id: "forest_zone",
    name: "Лесная зона",
    full_name: "Зона обитания единорогов в лесной местности",
    description: "Здесь единороги находят уединение и наслаждаются тенистыми местами с богатым растительным покровом."
  },
  {
    _id: "mount_zone",
    name: "Горная зона",
    full_name: "Зона обитания единорогов в горах",
    description: "Единороги здесь обретают высоту и наслаждаются прекрасными видами окружающей природы."
  },
  {
    _id: "polar_zone",
    name: "Полярная зона",
    full_name: "Зона обитания единорогов в полярных широтах",
    description: "Единороги в этой среде приспособлены к суровым условиям и наслаждаются бескрайними ледяными просторами."
  }
])

```

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

```

> db.unicorns.update(
  { _id: ObjectId("64953d68c6e023f3fe8b8c69") },
  {
    $set: {
      habitat_zone: {
        $ref: "unicorn_habitat_zones",
        $id: "polar_zone" }}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.update(
  { _id: ObjectId("64953d68c6e023f3fe8b8c6c") },
  {
    $set: {
      habitat_zone: {
        $ref: "unicorn_habitat_zones",
        $id: "forest_zone" }}})

```

3. Проверьте содержание коллекции единорогов.

```

> db.unicorns.find()
< {
  _id: ObjectId("64953d68c6e023f3fe8b8c69"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 73,
  habitat_zone: DBRef("unicorn_habitat_zones", 'polar_zone')
}
{
  _id: ObjectId("64953d68c6e023f3fe8b8c6a"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape',
    'sugar',
    'lemons'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43,
  habitat_zone: DBRef("unicorn_habitat_zones", 'mount_zone')
}

```

4. Содержание коллекции 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: 'Rooooooodles', 44), 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'});
```

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

Практическое задание 8.3.2:

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

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

2. Содержание коллекции единорогов *unicorns*:

```
db.unicorns.insert({name: 'Horny', dob:  
  new Date(1992,2,13,7,47), loves:  
  ['carrot','papaya'], weight: 600, gender: 'm',  
  vampires: 63});
```

```
db.unicorns.insert({name: 'Aurora', dob:  
  new Date(1991, 0, 24, 13, 0), loves: ['carrot',  
  'grape'], weight: 450, gender: 'f', vampires:  
  43});
```

```
db.unicorns.insert({name: 'Unicrom', dob:  
  new Date(1973, 1, 9, 22, 10), loves:  
  ['energon', 'redbull'], weight: 984, gender:  
  'm', vampires: 182});
```

```
db.unicorns.insert({name: 'Roooooodles',  
  dob: new Date(1979, 7, 18, 18, 44), loves:  
  ['apple'], weight: 575, gender: 'm',  
  vampires: 99});
```

```
db.unicorns.insert({name: 'Solnara', dob:  
  new Date(1985, 6, 4, 2, 1), loves:['apple',  
  'carrot', 'chocolate'], weight:550, gender:'f',  
  vampires:80});
```

```
db.unicorns.insert({name:'Ayna', dob: new  
  Date(1998, 2, 7, 8, 30), loves: ['strawberry',  
  'lemon'], weight: 733, gender: 'f', vampires:  
  40});
```

```
db.unicorns.insert({name:'Kenny',dob: new  
  Date(1997, 6, 1, 10, 42), loves: ['grape',  
  'lemon'], weight: 690, gender: 'm',  
  vampires: 39});
```

```
db.unicorns.insert({name: 'Raleigh', dob:  
  new Date(2005, 4, 3, 0, 57), loves: ['apple',  
  'sugar'], weight: 421, gender: 'm', vampires:  
  2});
```

```
db.unicorns.insert({name: 'Leia', dob: new Date(2001, 9, 8, 14, 53), loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
```

```
db.unicorns.insert({name: 'Pilot', dob: new Date(1997, 2, 1, 5, 3), loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
```

```
db.unicorns.insert ({name: 'Nimue', dob: new Date(1999, 11, 20, 16, 15), loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
```

```
db.unicorns.insert {name: 'Dunx', dob: new Date(1976, 6, 18, 18, 18), loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165
```

Практическое задание 8.3.3:

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

```
> 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.

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

3. Попробуйте удалить индекс для идентификатора.

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

Практическое задание 8.3.4:

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

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

```
> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6495539e1498e9670ca4707b")
  }
}
for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
```

2. Выберите последних четыре документа.

```
> db.numbers.find().sort({$natural: -1}).limit(4)
< {
  _id: ObjectId("6495539e1498e9670ca4707b"),
  value: 99999
}
{
  _id: ObjectId("6495539e1498e9670ca4707a"),
  value: 99998
}
{
  _id: ObjectId("6495539e1498e9670ca47079"),
  value: 99997
}
{
  _id: ObjectId("6495539e1498e9670ca47078"),
  value: 99996
}
```

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

```

> db.numbers.find().sort({$natural: -1}).limit(4).explain("executionStats")
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: {
        stage: 'COLLSCAN',
        direction: 'backward'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,

```

4. *Создайте индекс для ключа value.*

```

> db.numbers.createIndex({ value: 1 });
< value_1

```

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

```

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

```

6. *Выполните запрос 2.*


```
> db.numbers.find().sort({$natural: -1}).limit(4)
< {
  _id: ObjectId("6495539e1498e9670ca4707b"),
  value: 99999
}
{
  _id: ObjectId("6495539e1498e9670ca4707a"),
  value: 99998
}
{
  _id: ObjectId("6495539e1498e9670ca47079"),
  value: 99997
}
{
  _id: ObjectId("6495539e1498e9670ca47078"),
  value: 99996
}
}
```

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

```
> db.numbers.find().sort({$natural: -1}).limit(4).explain("executionStats")
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: {
        stage: 'COLLSCAN',
        direction: 'backward'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
  }
}
```

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

Обычно более эффективен будет запрос с индексами, так как он оперативнее ищет и работает с данными, имеющими конкретное значение. Однако на этих данных на моём компьютере всё отработало одинаково. Возможно, это произошло из-за мощности компьютера, который легко обрабатывает такой массив данных.

Вывод

В данной лабораторной работе было выполнено ознакомление с MongoDB, его командами. Была осуществлена работа с несколькими коллекциями базы данных, совершалось добавление элементов, их обновление и удаление, работа с ограничением вывода. Была создана коллекция, на которую совершались ссылки с другой коллекции. Осуществлено добавление, настройка и удаление индексов, анализ поиска по индексу и без него.