**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет инфокоммуникационных технологий**

**Дисциплина:**
«Проектирование и реализация баз данных»

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
# «РЕАЛИЗАЦИЯ БД С ИСПОЛЬЗОВАНИЕМ MONGO DB. ЗАПРОСЫ К БАЗАМ ДАННЫХ»

**Выполнил:**
студент группы K32391
Микитчак Иван Михайлович

_____
(подпись)

**Проверил:**
Говорова Марина Михайловна

_____
(отметка о выполнении)

_____
(подпись)

Санкт-Петербург
202 г.

**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.

**Ход выполнения**

Практическое задание 8.1.1

1) Создать базу данных unicorns

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

2) Вставить данные

```
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("648f7d1f8e7daab3803300fc") }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d1f8e7daab3803300fd") }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d1f8e7daab3803300fe") }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d1f8e7daab3803300ff") }
}
learn> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d1f8e7daab380330100") }
}
learn> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d1f8e7daab380330101") }
}
learn> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d1f8e7daab380330102") }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d208e7daab380330103") }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d208e7daab380330104") }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d208e7daab380330105") }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7d208e7daab380330106") }
}
learn>
```

3) Используя второй способ, вставьте документ в коллекцию

```
learn> document=({name:'Dunx', loves:['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document)
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("648f7f6db205aa4ae51ec83c") }
}
learn>
```

4) Проверьте содержимое коллекции

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId("648f7d1f8e7daab3803300fc"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("648f7d1f8e7daab3803300fd"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("648f7d1f8e7daab3803300fe"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("648f7d1f8e7daab3803300ff"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("648f7d1f8e7daab380330100"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("648f7d1f8e7daab380330101"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId("648f7d1f8e7daab380330102"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
```

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("648f7d1f8e7daab3803300fd"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("648f7d1f8e7daab380330101"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId("648f7d208e7daab380330104"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn>
```

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId("648f7f6db205aa4ae51ec83c"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("648f7d1f8e7daab3803300fc"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("648f7d1f8e7daab380330102"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
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'}).limit(1)
[
  {
    _id: ObjectId("648f7d1f8e7daab3803300fd"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

Практическое задание 8.1.3

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
[
  {
    _id: ObjectId("648f7d1f8e7daab3803300fc"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId("648f7d1f8e7daab3803300fe"),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId("648f7d1f8e7daab3803300ff"),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId("648f7d1f8e7daab380330102"),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId("648f7d208e7daab380330103"),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId("648f7d208e7daab380330105"),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId("648f7f6db205aa4ae51ec83c"),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
]
learn>
```

Практическое задание 8.1.4

Вывести список единорогов в обратном порядке добавления.


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' ],
```

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: {$gte: 500, $lte: 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: {$gte: 500}, loves: {$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({vampires: {$exists: 0}})
[
  {
    _id: ObjectId("648f7d208e7daab380330106"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
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, weight: 0, gender: 0, vampires: 0, 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' ] }
]
learn>
```

Практическое задание 8.2.1

1) Создайте коллекцию towns, включающую указанные документы

```
learn> db.towns.find()
[
  {
    _id: ObjectId("64906849108a3296d97037c5"),
    name: 'Punxsutawney',
    population: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId("649068bf108a3296d97037c6"),
    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("649068db108a3296d97037c7"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
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, populatiuon: 0, last_sensus: 0, famous_for: 0})
[
  {
    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: 0}}, {_id: 0, populatiuon: 0, last_sensus: 0, famous_for: 0})
[ { name: 'Punxsutawney ', mayor: { name: 'Jim Wehrle' } } ]
learn>
```

Практическое задание 8.2.2

1) Сформировать функцию для вывода списка самцов единорогов.

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

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

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

3) Вывести результат, используя forEach.

```
learn> cursor.forEach(function(obj) {print(obj)})
{
  _id: ObjectId("648f7f6db205aa4ae51ec83c"),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId("648f7d1f8e7daab3803300fc"),
  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: {$gte: 500, $lte: 600}}).count()
2
learn>
```

Практическое задание 8.2.4

Вывести список предпочтений.

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

Практическое задание 8.2.5

Посчитать количество особей единорогов обоих полов.

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

Практическое задание 8.2.6

1) Выполнить команду из условия

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.replaceOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
learn>
```

```
learn> db.unicorns.insert({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6490b423f1da4ae255be1a90") }
}
learn>
```

Практическое задание 8.2.7

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

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>
```

Практическое задание 8.2.8

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

2) Проверить содержимое коллекции unicorns.

Проверка ниже в пункте 8.2.9

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

Практическое задание 8.2.9

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

2) Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
learn>
```

```
learn> db.unicorns.find({gender: 'm'})
[
  {
    _id: ObjectId("648f7d1f8e7daab3803300fc"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId("648f7d1f8e7daab3803300fe"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId("648f7d1f8e7daab3803300ff"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104
learn>
  {
    _id: ObjectId("648f7d1f8e7daab380330102"),
    name: 'Kenny',
```

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> _
```

```
learn> db.towns.find()
[
  {
    _id: ObjectId("649069ce108a3296d97037c8"),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId("649069e3108a3296d97037c9"),
    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("649069f7108a3296d97037ca"),
    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("648f7d208e7daab380330105"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
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', 'lemon']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Aurora'})
[
  {
    _id: ObjectId("648f7d1f8e7daab3803300fd"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

Практическое задание 8.2.13

1) Создайте коллекцию towns, включающую указанные документы

```
learn> db.towns.find()
[
  {
    _id: ObjectId("649069ce108a3296d97037c8"),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ 'phil the groundhog' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId("649069e3108a3296d97037c9"),
    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("649069f7108a3296d97037ca"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>
```

2) Удалите документы с беспартийными мэрами.

3) Проверьте содержание коллекции.

```
learn> db.towns.remove({'mayor.party': {$exists: 0}})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId("649069e3108a3296d97037c9"),
    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("649069f7108a3296d97037ca"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>
```

4) Очистите коллекцию.

5) Просмотрите список доступных коллекций.

```
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.habitat.find()
[
  {
    _id: 'GM',
    name: 'Grassy Meadow',
    description: 'A peaceful grassy meadow'
  },
  { _id: 'EW', name: 'Empty Wastes', description: 'A grim place' },
  {
    _id: 'MF',
    name: 'Mysterious Forest',
    description: 'An ancient forest'
  }
]
learn>
```

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

```
learn> db.unicorns.updateOne({name: 'Barny'}, {$set: {habitat: {$ref: 'habitat', $id: 'GM'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne({name: 'Dunx'}, {$set: {habitat: {$ref: 'habitat', $id: 'EW'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne({name: 'Aurora'}, {$set: {habitat: {$ref: 'habitat', $id: 'MF'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>
```

3) Проверьте содержание коллекции едиорогов.

```
learn> db.unicorns.find({$or: [{name: 'Aurora'}, {name: 'Dunx'}, {name: 'Barny'}]})
[
  {
    _id: ObjectId("648f7d1f8e7daab3803300fd"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef("habitat", 'MF')
  },
  {
    _id: ObjectId("648f7f6db205aa4ae51ec83c"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 170,
    habitat: DBRef("habitat", 'EW')
  },
  {
    _id: ObjectId("6490b423f1da4ae255be1a90"),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm',
    vampires: 5,
    habitat: DBRef("habitat", 'GM')
  }
]
```

Практическое задание 8.3.2

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

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

Практическое задание 8.3.3

1) Получите информацию о всех индексах коллекции unicorns .

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

2) Удалите все индексы, кроме индекса для идентификатора.

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

3) Попытайтесь удалить индекс для идентификатора.

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

Практическое задание 8.3.4

1) Создайте объемную коллекцию numbers, задействовав курсор:

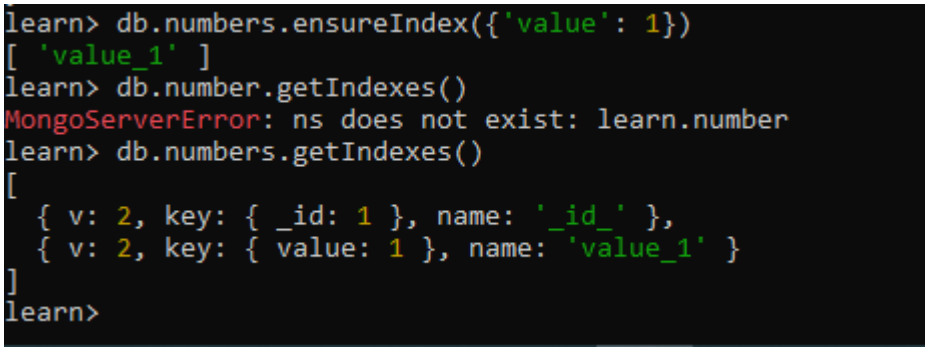
```
learn> db.createCollection('numbers')
{ ok: 1 }
learn> for(i=0; i<100000; i++) {db.numbers.insert({value: i})}
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6490cb7139b3c748347031c2") }
}
learn>
```

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? (запрос с индексом)

```
learn> db.numbers.explain('executionStats').find().sort({value: -1}).limit(4)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: 'B0086AC2',
    planCacheKey: 'B0086AC2',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: {
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { value: 1 },
          indexName: 'value_1',
          isMultiKey: false,
          multiKeyPaths: { value: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'backward',
          indexBounds: { value: [ '[MaxKey, MinKey]' ] }
        }
      }
    },
    rejectedPlans: []
  },
```

```
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 2,
    totalKeysExamined: 4,
    totalDocsExamined: 4,
    executionStages: {
      stage: 'LIMIT',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      limitAmount: 4,
      inputStage: {
        stage: 'FETCH',
        nReturned: 4,
        executionTimeMillisEstimate: 0,
        works: 4,
        advanced: 4,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 0,
        docsExamined: 4,
        alreadyHasObj: 0,
        inputStage: {
          stage: 'IXSCAN',
          nReturned: 4,
          executionTimeMillisEstimate: 0,
          works: 4,
          advanced: 4,
          needTime: 0,
          needYield: 0,
          saveState: 0,
          restoreState: 0,
          isEOF: 0,
          keyPattern: { value: 1 },
          indexName: 'value_1',
          isMultiKey: false,
          multiKeyPaths: { value: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'backward',
          indexBounds: { value: [ '[MaxKey, MinKey]' ] },
          keysExamined: 4,
          seeks: 1,
          dupsTested: 0,
          dupsDropped: 0
        }
      }
    }
  },
```

## Вывод:

В ходе работы были овладены 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.