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

Отчет

по лабораторной работе №6 «Работа с БД в СУБД MongoDB**»**

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

Автор: Григорьев А. В.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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

Оборудование: компьютерный класс.

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

Выполнение

Практическое задание 2.1.1:

1) Создайте базу данных learn.

2) Заполните коллекцию единорогов unicorns:

```
test> use learn
switched to db learn
learn> db.createCollecion('unicorns')
TypeError: db.createCollecion is not a function
learn> db
learn
learn> db.createCollecion("unicorns")
TypeError: db.createCollecion is not a function
learn> db.createCollection("unicorns")
{ ok: 1 }
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('658563cf36904a19e6b7b850') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 4
);
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('658563cf36904a19e6b7b851') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampire
 182});
elon'], weight: 650, gender: 'm', vampires: 54});
db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('658563cf36904a19e6b7b852') }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('658563cf36904a19e6b7b853') }
}
learn> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f',
mpires:80});
{
  acknowledged: true,
```

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

```
learn> unicorn = ({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(unicorn)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6585649536904a19e6b7b85b') }
}
```

4) Проверьте содержимое коллекции с помощью метода find.

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('658563cf36904a19e6b7b850'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b851'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b852'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b853'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b854'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b855'),
```

Практическое задание 2.2.1:

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

Список самцов:

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('6585649536904a19e6b7b85b'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b850'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b856'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b859'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b857'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b853'),
    name: 'Roooooodles',
```

Список самок:

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('658563cf36904a19e6b7b851'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b855'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b858'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

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

findOne:

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId('658563cf36904a19e6b7b851'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Limit:

```
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId('658563cf36904a19e6b7b851'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 2.2.2:

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).sort({name: 1})
[
  {
    _id: ObjectId('6585649536904a19e6b7b85b'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b850'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b856'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b859'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b857'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b853'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b852'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
]
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('6585649536904a19e6b7b85b'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b85a'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b859'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b858'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b857'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b856'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
```

Практическое задание 2.1.4

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

```
learn> db.unicorns.find({}, {loves: {$slice: 1}, _id: 0})
[
  {
    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 2.3.1

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'
  }
]
```

Практическое задание 2.3.2

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
  }
]
```

Практическое задание 2.3.3

Практическое задание 2.3.4

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: {$slice: 1}}).sort({name: 1})
[
  {
    _id: ObjectId('6585649536904a19e6b7b85b'),
    name: 'Dunx',
    loves: [ 'grape' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b850'),
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b856'),
    name: 'Kenny',
    loves: [ 'grape' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b859'),
    name: 'Pilot',
    loves: [ 'apple' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b857'),
    name: 'Raleigh',
    loves: [ 'apple' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b853'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
```

Практическое задание 3.1.1

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

```
learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     population: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: [""],
...     mayor: {
...       name: "Jim Wehrle"
...     }
...   },
...   {
...     name: "New York",
...     population: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["Statue 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('6585930536904a19e6b7b85c'),
    '1': ObjectId('6585930536904a19e6b7b85d'),
    '2': ObjectId('6585930536904a19e6b7b85e')
  }
}
```

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'}, {name: 1, 'mayor.name': 1, _id: 0})
[ { 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.

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

Практическое задание 3.1.2

4) Сформировать функцию для вывода списка самцов единорогов.

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

6) Вывести результат, используя forEach.

```
learn> printUnicorns = () => {let cursor = db.unicorns.find({gender: 'm'}); cursor.sort({name: 1}).limit(2); c
ursor.forEach((unicorn) => {console.log(unicorn.name)})}
[Function: printUnicorns]
learn> printUnicorns
[Function: printUnicorns]
learn> printUnicorns()
Dunx
Horny
```

Практическое задание 3.2.1

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
learn> db.unicorns.find({gender: 'm', weight: {$gte: 500, $lte: 600}}).count()
(node:69302) [MONGODB DRIVER] Warning: cursor.count is deprecated and will be removed in the next major versi
n, please use `collection.estimatedDocumentCount` or `collection.countDocuments` instead
2
```

Практическое задание 3.2.2

Вывести список предпочтений.

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

Практическое задание 3.2.3

Посчитать количество особей единорогов обоих полов

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

Практическое задание 3.3.1

1) Выполнить команду:

> db.unicorns.save({name: 'Barny', loves: ['grape'],

weight: 340, gender: 'm'})

2) Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedId: ObjectId('6585d03936904a19e6b7b85f')
}
```

```
  {
    _id: ObjectId('6585d03936904a19e6b7b85f'),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
```

Практическое задание 3.3.2

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

```
learn> db.unicorns.update({name: 'Ayna'}, {$set: {weight: 800, vampires: 51}})
[DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
    _id: ObjectId('658563cf36904a19e6b7b855'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  },
```

Практическое задание 3.3.3

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

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

```
  {
    _id: ObjectId('658563cf36904a19e6b7b857'),
    name: 'Raleigh',
    loves: 'redbull',
    weight: 421,
    gender: 'm',
    vampires: 2
  },
```

Практическое задание 3.3.4

Всем самцам единорогов увеличить количество убитых вапмиров на 5.

```
learn> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('658563cf36904a19e6b7b850'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 78
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b851'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b852'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
```

Практическое задание 3.3.5

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

```
learn> db.towns.update({name: 'Portland'}, {$set: {'mayor.party': 'I'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({name: Portland})
ReferenceError: Portland is not defined
learn> db.towns.find({name: 'Portland'})
[
  {
    _id: ObjectId('6585930536904a19e6b7b85e'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'I' }
  }
]
```

Практическое задание 3.3.6

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

```
learn> db.unicorns.update({name: 'Pilot'}, {$push: {loves: 'chocolate'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Pilot'})
[
  {
    _id: ObjectId('658563cf36904a19e6b7b859'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

Практическое задание 3.3.7

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

```
learn> db.unicorns.update({name: 'Aurora'}, {$push: {loves: {$each: ['sugar', 'lemon']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: "Aurora"})
[
  {
    _id: ObjectId('658563cf36904a19e6b7b851'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 3.4.1

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.

1. *Проверьте содержание коллекции.*
2. *Очистите коллекцию.*
3. *Просмотрите список доступных коллекций.*

```
learn> db.towns.deleteMany({'mayor.party': {$exists: false}})
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId('6585930536904a19e6b7b85d'),
    name: 'New York',
    population: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'Statue of Liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('6585930536904a19e6b7b85e'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

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

Практическое задание 4.1.1

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

3) Включите для нескольких единорогов в документы ссылку на зону обитания, использую второй способ автоматического связывания.
4) Проверьте содержание коллекции едиорогов.
5) Содержание коллекции единорогов unicorns:

```
learn> db.areas.insertMany([ {_id: 'to', name: 'Tihiy Ocean', descr: 'The biggest yacht' }, {_id: 'sp', name:
'saint p.', descr: 'The biggest swamp' }, {_id: 'mo', name: 'moscow', descr: 'The capital of the country' }]
{
  acknowledged: true,
  insertedIds: { '0': 'to', '1': 'sp', '2': 'mo' }
}
```

```
learn> db.unicorns.updateMany({gender: 'f'}, {$set: {area: {$ref: 'areas', $id: 'sp'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 5,
  modifiedCount: 5,
  upsertedCount: 0
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('658563cf36904a19e6b7b850'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 78
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b851'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    area: DBRef('areas', 'sp')
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b852'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId('658563cf36904a19e6b7b853'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
```

Практическое задание 4.2.1

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

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

learn> db.unicorns.getIndexes()
```

Практическое задание 4.3.1

1. *Получите информацию о всех индексах коллекции* `unicorns`.
2. *Удалите все индексы, кроме индекса для идентификатора.*
3. *Попытайтесь удалить индекс для идентификатора.*

```
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> db.unicorns.dropIndexes()
{
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
```

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

Практическое задание 4.4.1

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

```
numbers> for (i = 0; i < 100000; i++) {db.numbers.insert({value: i})}
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6589720a36904a19e6bc0cbd') }
}
```

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?*

```
numbers> db.numbers.find().sort({value: -1}).limit(4).explain('executionStats')
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'numbers.numbers',
    indexFilterSet: false,
```

```
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 70,
    totalKeysExamined: 0,
    totalDocsExamined: 183739,
    executionStages: {
```

```
numbers> db.numbers.find().sort({value: -1}).limit(4).explain('executionStats')
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'numbers.numbers',
    indexFilterSet: false,
    parsedQuery: {},
```

```
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 9,
    totalKeysExamined: 4,
    totalDocsExamined: 4,
    executionStages: {
```

Время выполнения без индекса: 70 мс

Время выполнения с индексом: 4 мс
Время выполнения с индексом превосходит в 17.5 раза.

Вывод

В ходе лабораторной работы была освоена работа с СУБД MongoDB. Были проведены практические работы с CRUD-операциями, вложенными объектами, агрегациями, изменениями данных, ссылками и индексами.