МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

Факультет Прикладная Информатика

ЛАБОРАТОРНАЯ РАБОТА № 6

по дисциплине

“ Проектирование и реализация баз данных”

Выполнил:
Джафари Хоссаин

Студент группы K3240

Преподаватель:
Белов Александр Олегович
Говорова Марина Михайловна

Санкт-Петербург, 2025

Contents

Цель работы

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 8.0.4 (последняя).

Выполнение

Практическое задание:

Практическое задание 2.1.1:

1. Создайте базу данных learn.
2. Заполните коллекцию единорогов unicorns:

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

```
> db.unicorns.insertMany([
   {name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63},
   {name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43},
   {name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182},
   {name: 'Roooooodles', 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('684c2eafe249428ec7ca1286'),
    '1': ObjectId('684c2eafe249428ec7ca1287'),
    '2': ObjectId('684c2eafe249428ec7ca1288'),
    '3': ObjectId('684c2eafe249428ec7ca1289'),
    '4': ObjectId('684c2eafe249428ec7ca128a'),
    '5': ObjectId('684c2eafe249428ec7ca128b'),
    '6': ObjectId('684c2eafe249428ec7ca128c'),
    '7': ObjectId('684c2eafe249428ec7ca128d'),
    '8': ObjectId('684c2eafe249428ec7ca128e'),
    '9': ObjectId('684c2eafe249428ec7ca128f'),
    '10': ObjectId('684c2eafe249428ec7ca1290')
  }
}
```

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

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

```
> db.unicorns.insertOne({
    name: 'Dunx',
    loves: ['grape', 'watermelon'],
    weight: 704,
    gender: 'm',
    vampires: 165
  });
< {
    acknowledged: true,
    insertedId: ObjectId('684c2f50e249428ec7ca1291')
  }
```

4. Проверьте содержимое коллекции с помощью метода find.

```
> db.unicorns.find()
< {
    _id: ObjectId('684c2eafe249428ec7ca1286'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('684c2eafe249428ec7ca1287'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
```

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:

```
> db.unicorns.find({gender: "m"}).sort({name: 1})
< {
    _id: ObjectId('684c2f50e249428ec7ca1291'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('684c2eafe249428ec7ca1286'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
```

Список самок:

```
> db.unicorns.find({gender: "f"}).sort({name: 1}).limit(3)
< {
    _id: ObjectId('684c2eafe249428ec7ca1287'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('684c2eafe249428ec7ca128b'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
    _id: ObjectId('684c2eafe249428ec7ca128e'),
    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:

```
> db.unicorns.findOne({ gender: "f", loves: "carrot" });
< {
    _id: ObjectId('684c2eafe249428ec7ca1287'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

Limit:

```
> db.unicorns.find({ gender: "f", loves: "carrot" }).limit(1);
< {
    _id: ObjectId('684c2eafe249428ec7ca1287'),
    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 предпочтениях и поле.

```
> db.unicorns.find({gender: "m"},{gender: false, loves: false})
< {
    _id: ObjectId('684c2eafe249428ec7ca1286'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
  {
    _id: ObjectId('684c2eafe249428ec7ca1288'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
  {
    _id: ObjectId('684c2eafe249428ec7ca1289'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  }
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find().sort({_id: -1});
< {
    _id: ObjectId('684c2f50e249428ec7ca1291'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('684c2eafe249428ec7ca1290'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
```

Практическое задание 2.1.4:

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

```
> db.unicorns.find({}, {_id:0, name: 1, loves: {$slice: 1}}).pretty();
< {
  name: 'Horny',
  loves: [
    'carrot'
  ]
}
{
  name: 'Aurora',
  loves: [
    'carrot'
  ]
}
{
  name: 'Unicrom',
  loves: [
    'energon'
  ]
}
```

Практическое задание 2.3.1:

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

```
> 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',
```

Практическое задание 2.3.2:

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

```
> 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
  }
```

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

```
> db.unicorns.find({vampires: {$exists: false}})
< {
    _id: ObjectId('684c2eafe249428ec7ca1290'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

Практическое задание 2.3.4:

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

```
> db.unicorns.find({ gender: "m" },{_id: 0, name: 1, loves: { $slice: 1 }}).sort({ name: 1 })
< {
    name: 'Dunx',
    loves: [
      'grape'
    ]
  }
  {
    name: 'Horny',
    loves: [
      'carrot'
    ]
  }
  {
    name: 'Kenny',
    loves: [
      'grape'
    ]
  }
```

Практическое задание 3.1.1:

1. Создайте коллекцию towns, включающую следующие документы:

```
> 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('684c613de249428ec7ca1292'),
      '1': ObjectId('684c613de249428ec7ca1293'),
      '2': ObjectId('684c613de249428ec7ca1294')
    }
  }
```

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

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

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

Практическое задание 3.1.2:

1. Сформировать функцию для вывода списка самцов единорогов.

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

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

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

3. Вывести результат, используя forEach

```
> db.unicorns.find().sort({ name: 1 }).limit(2).forEach(function(unicorn) {
    print(unicorn.name);
  });
< Aurora
< Ayna
```

4. Содержание коллекции единорогов unicorns

```
> db.unicorns.find({$where : fn})
< {
    _id: ObjectId('684c2eafe249428ec7ca1286'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('684c2eafe249428ec7ca1288'),
    name: 'Unicrom',
    loves: [
      'energon',
      'redbull'
    ],
```

```
> 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: 1
  db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
  db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vamp
  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
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('684c6f7de249428ec7ca12a0')
    }
  }
```

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
> db.unicorns.find({gender:"f"} , {weight : {$gte : 500 ,$lte : 600}}).count();
< 10
```

Практическое задание 3.2.2:

Вывести список предпочтений.

```
> 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.

```
> db.unicorns.aggregate([{ "$group": { _id: "$gender", count: { $sum: 1 } } }]);
< {
    _id: 'm',
    count: 14
  }
  {
    _id: 'f',
    count: 10
  }
```

Практическое задание 3.3.1:

1. Выполнить команду:

```
> db.unicorns.updateOne(
    { name: "Barny" },  // фильтр
    {
      $set: {
        loves: ["grape"],
        weight: 340,
        gender: "m"
      }
    },
    { upsert: true }  // если не найдено — создать
  )
< {
    acknowledged: true,
    insertedId: ObjectId('684c92a2e4c0455f2a0538ea'),
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 1
  }
```

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.find();
```

```
  _id: ObjectId('684c6f7de249428ec7ca12a0'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('684c92a2e4c0455f2a0538ea'),
  name: 'Barny',
  gender: 'm',
  loves: [
    'grape'
  ],
  weight: 340
}
```

Практическое задание 3.3.2:

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

```
> db.unicorns.updateOne({ name: "Ayna", gender: "f" },{ $set: {weight: 800,vampires: 51}});
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.find();
```

Практическое задание 3.3.3:

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

```
> db.unicorns.updateOne(
    { name: "Raleigh", gender: "m" },
    { $addToSet: { loves: "redbull" } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.find();
```

```
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
{
  _id: ObjectId('684c6f7de249428ec7ca129c'),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
```

Практическое задание 3.3.4:

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

```
> db.unicorns.updateMany(
    { gender: "m" },
    { $inc: { vampires: 5 } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 15,
    modifiedCount: 15,
    upsertedCount: 0
  }
```

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.find();
```

```
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 44
}
{
  _id: ObjectId('684c6f7de249428ec7ca129c'),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar'
  ],
  weight: 421,
  gender: 'm',
  vampires: 7
}
```

Практическое задание 3.3.5:

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

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

2. Проверить содержимое коллекции towns.

```
> db.towns.find({ name: "Portland" }).pretty();
< {
    _id: ObjectId('684c613de249428ec7ca1294'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [
      'beer',
      'food'
    ],
    mayor: {
      name: 'Sam Adams'
    }
  }
```

Практическое задание 3.3.6:

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

```
> db.unicorns.updateOne({ name: "Pilot" }, { $addToSet: { loves: "chocolate" } });
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.find({ name: "Pilot" }).pretty();
< {
    _id: ObjectId('684c2eafe249428ec7ca128f'),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon',
      'chocolate'
    ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
```

Практическое задание 3.3.7:

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

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

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.find({ name: "Aurora" }).pretty();
< {
    _id: ObjectId('684c2eafe249428ec7ca1287'),
    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"}}

```
> db.towns.insertMany([
    {
      name: "Punxsutawney",
      population: 6200,
      last_sensus: ISODate("2008-01-31"),
      famous_for: ["phil the groundhog"],
      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('684de6677209648cd759b883'),
      '1': ObjectId('684de6677209648cd759b884'),
      '2': ObjectId('684de6677209648cd759b885')
    }
  }
```

2. Удалите документы с беспартийными мэрами.

```
> db.towns.deleteMany({ "mayor.party": { $exists: false } });
< {
    acknowledged: true,
    deletedCount: 3
  }
```

3. Проверьте содержание коллекции.

```
> db.towns.find().pretty()
< {
    _id: ObjectId('684c613de249428ec7ca1293'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [
      'status of liberty',
      'food'
    ],
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
```

4. Очистите коллекцию.

```
> db.towns.deleteMany({});
< {
    acknowledged: true,
    deletedCount: 3
  }
```

5. Просмотрите список доступных коллекций

```
> 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.

```
> db.habitats.insertMany([
    {
      _id: "NF",
      name: "Northern Forest",
      description: "A dense forest with towering pines and magical glades, often misty."
    },
    {
      _id: "DS",
      name: "Desert Sands",
      description: "A vast desert region with rare oases, inhabited by heat-tolerant unicorns."
    },
    {
      _id: "MV",
      name: "Mystic Valley",
      description: "A hidden valley filled with enchanted flora and glowing waterfalls."
    }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': 'NF',
      '1': 'DS',
      '2': 'MV'
    }
  }
```

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

```
learn> db.unicorns.updateOne(
         { name: "Aurora" },
         { $set: { habitat_id: "NF" } }
       )

       db.unicorns.updateOne(
         { name: "Unicrom" },
         { $set: { habitat_id: "DS" } }
       )

       db.unicorns.updateOne(
         { name: "Leia" },
         { $set: { habitat_id: "MV" } }
       );
```

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

3. Проверьте содержание коллекции едиорогов.

```
> db.unicorns.find().pretty()
< {
    _id: ObjectId('684c2eafe249428ec7ca1286'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 68
  }
  {
    _id: ObjectId('684c2eafe249428ec7ca1287'),
    name: 'Aurora',
    loves: [
      'carrot',
```

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: 1
  db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
  db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vamp
  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
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('684df20c7209648cd759b891')
    }
  }
```

Практическое задание 4.2.1:

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

```
> db.unicorns.createIndex({name: 1}, {unique: true})
< name_1
```

2. Содержание коллекции единорогов unicorns:

```
> db.unicorns.insertMany([
    {name: 'Horny', dob: new Date(1992, 2, 13, 7, 47), loves: ['carrot','papaya'], weight: 600, gender: 'm'
    {name: 'Aurora', dob: new Date(1991, 0, 24, 13, 0), loves: ['carrot', 'grape'], weight: 450, gender: 'f
    {name: 'Unicrom', dob: new Date(1973, 1, 9, 22, 10), loves: ['energon', 'redbull'], weight: 984, gender
    {name: 'Roooooodles', dob: new Date(1979, 7, 18, 18, 44), loves: ['apple'], weight: 575, gender: 'm',
    {name: 'Solnara', dob: new Date(1985, 6, 4, 2, 1), loves: ['apple', 'carrot', 'chocolate'], weight: 550
    {name: 'Ayna', dob: new Date(1998, 2, 7, 8, 30), loves: ['strawberry', 'lemon'], weight: 733, gender:
    {name: 'Kenny', dob: new Date(1997, 6, 1, 10, 42), loves: ['grape', 'lemon'], weight: 690, gender: 'm'
    {name: 'Raleigh', dob: new Date(2005, 4, 3, 0, 57), loves: ['apple', 'sugar'], weight: 421, gender: 'm'
    {name: 'Leia', dob: new Date(2001, 9, 8, 14, 53), loves: ['apple', 'watermelon'], weight: 601, gender:
    {name: 'Pilot', dob: new Date(1997, 2, 1, 5, 3), loves: ['apple', 'watermelon'], weight: 650, gender:
    {name: 'Nimue', dob: new Date(1999, 11, 20, 16, 15), loves: ['grape', 'carrot'], weight: 540, gender:
    {name: 'Dunx', dob: new Date(1976, 6, 18, 18, 18), loves: ['grape', 'watermelon'], weight: 704, gender
  ]);
```

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('684df4517209648cd759b894'),
    '1': ObjectId('684df4517209648cd759b895'),
    '2': ObjectId('684df4517209648cd759b896'),
    '3': ObjectId('684df4517209648cd759b897'),
    '4': ObjectId('684df4517209648cd759b898'),
    '5': ObjectId('684df4517209648cd759b899'),
    '6': ObjectId('684df4517209648cd759b89a'),
    '7': ObjectId('684df4517209648cd759b89b'),
    '8': ObjectId('684df4517209648cd759b89c'),
    '9': ObjectId('684df4517209648cd759b89d'),
    '10': ObjectId('684df4517209648cd759b89e'),
    '11': ObjectId('684df4517209648cd759b89f')
  }
}
```

Практическое задание 4.3.1:

1. Получите информацию о всех индексах коллекции unicorns .

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

2. Удалите все индексы, кроме индекса для идентификатора

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

3. Попытайтесь удалить индекс для идентификатора.

```
> db.unicorns.dropIndex("_id_")
⊗ ▸ MongoServerError[InvalidOptions]: cannot drop _id index
```

Практическое задание 4.4.1:

1. Создайте объемную коллекцию numbers, задействовав курсор:

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

```
> for (let i = 0; i < 100000; i++) {
    db.numbers.insert({ value: i });
  }
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('684df6827209648cd75b3f3f')
    }
  }
```

2. Выберите последних четыре документа.

```
> db.numbers.find().sort({ value: -1 }).limit(4)
< {
    _id: ObjectId('684df6827209648cd75b3f3f'),
    value: 99999
  }
  {
    _id: ObjectId('684df6827209648cd75b3f3e'),
    value: 99998
  }
  {
    _id: ObjectId('684df6827209648cd75b3f3d'),
    value: 99997
  }
  {
    _id: ObjectId('684df6827209648cd75b3f3c'),
    value: 99996
  }
learn>
```

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

```
executionTimeMillis: 138,
```

4. Создайте индекс для ключа value.

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

5. Получите информацию о всех индексах коллекции numbres.

```
> 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({ value: -1 }).limit(4)
< {
    _id: ObjectId('684df6827209648cd75b3f3f'),
    value: 99999
  }
  {
    _id: ObjectId('684df6827209648cd75b3f3e'),
    value: 99998
  }
  {
    _id: ObjectId('684df6827209648cd75b3f3d'),
    value: 99997
  }
  {
    _id: ObjectId('684df6827209648cd75b3f3c'),
    value: 99996
  }
```

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

```
executionTimeMillis: 0,
```

Вывод :

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