

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

Отчет

по лабораторной работе №6 «Создание таблиц базы данных PostgreSQL. Заполнение таблиц
рабочими данными»

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

Автор: Ковалев Г. П.

Факультет: ПИН

Группа: К3241

Преподаватель: Говорова М.М.



Санкт-Петербург 2025

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.

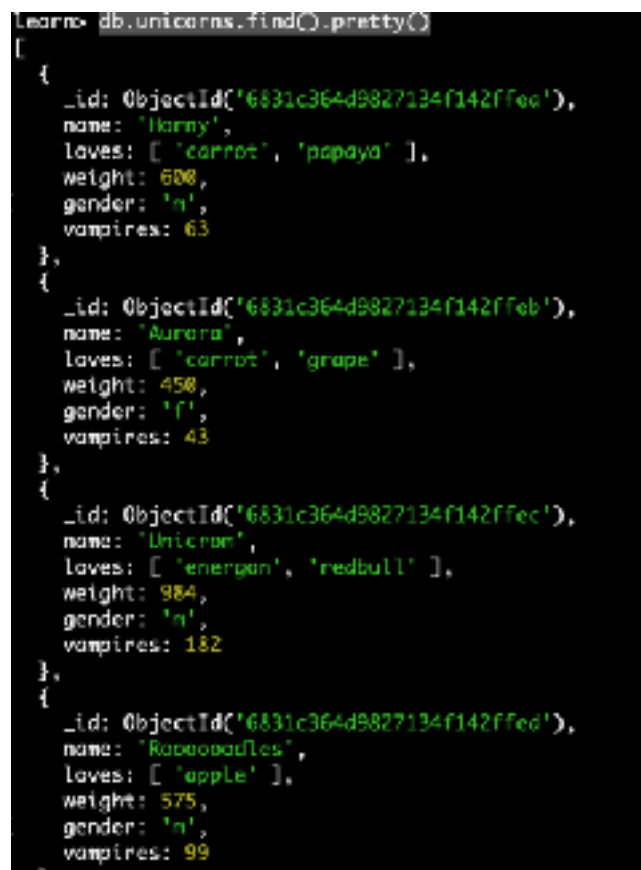
2.1.1) Создайте базу данных learn. Заполните коллекцию единорогов unicorns. Используя второй способ, вставьте в коллекцию единорогов документ. Проверьте содержимое коллекции с помощью метода find.

use learn

```
db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
db.unicorns.insert({name: 'Rooooooodles', 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.insertOne({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165});

db.unicorns.find().pretty()
```



```
Learn> db.unicorns.find().pretty()
[
  {
    _id: ObjectId('6831c364d9827134f142fFeb'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6831c364d9827134f142fFeb'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6831c364d9827134f142fFeb'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('6831c364d9827134f142fFeb'),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  }
]
```

2.2.1) Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени. Найдите всех самок, которые любят carrot. Ограничьте этот список первой особью с помощью функций findOne и limit.

1.1) db.unicorns.find({gender: 'm'}).sort({name: 1})

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('6831c364d9827134f142ff5'),
    name: 'Duke',
    loves: [ 'pepe', 'watermelon' ],
    weight: 724,
    gender: 'm',
    vampires: 20
  },
  {
    _id: ObjectId('6831c364d9827134f142ff6'),
    name: 'Henry',
    loves: [ 'carrot', 'pepeys' ],
    weight: 686,
    gender: 'm',
    vampires: 13
  },
  {
    _id: ObjectId('6831c364d9827134f142ff7'),
    name: 'Lenny',
    loves: [ 'pepe', 'lemon' ],
    weight: 696,
    gender: 'm',
    vampires: 14
  },
  {
    _id: ObjectId('6831c364d9827134f142ff8'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 652,
    gender: 'm',
    vampires: 14
  },
  {
    _id: ObjectId('6831c364d9827134f142ff9'),
    name: 'Kathryn',
    loves: [ 'apple', 'caper' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('6831c364d9827134f142ffa'),
    name: 'Reeseed:es',
    loves: [ 'apple' ],
    weight: 573,
    gender: 'm',
    vampires: 20
  },
  {
    _id: ObjectId('6831c364d9827134f142ffb'),
    name: 'Mikaela',
    loves: [ 'amargen', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

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

```
learn> db.unicorns.find({gender: 'F'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('6831c364d9827134f142ffeb'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'F',
    vampires: 43
  },
  {
    _id: ObjectId('6831c364d9827134f142ffef'),
    name: 'Aynd',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'F',
    vampires: 48
  },
  {
    _id: ObjectId('6831c364d9827134f142fff2'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'F',
    vampires: 33
  }
]
```

- 2.1) db.unicorns.findOne({gender: 'f', loves: 'carrot'})
 2.2) db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId('6831c364d9827134f142ffeb'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId('6831c364d9827134f142ffeb'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

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

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).sort({name: 1})
[
  {
    _id: ObjectId('6831c386d9827134f142ff45'),
    name: 'Euan',
    weight: 789,
    vampires: 118
  },
  {
    _id: ObjectId('6831c364d9827134f142ffeb'),
    name: 'Horny',
    weight: 640,
    vampires: 63
  },
  {
    _id: ObjectId('6831c364d9827134f142ff40'),
    name: 'Henry',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('6831c364d9827134f142ff43'),
    name: 'Pilot',
    weight: 690,
    vampires: 54
  },
  {
    _id: ObjectId('6831c364d9827134f142ff41'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('6831c364d9827134f142ffed'),
    name: 'Rumorsales',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('6831c364d9827134f142ffec'),
    name: 'Thecrow',
    weight: 944,
    vampires: 192
  }
]
```

2.2.3) Вывести список единорогов в обратном порядке добавления.

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

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('6831c360d9827134f142fff5'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6831c364d9827134f142fff4'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6831c364d9827134f142fff3'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
]
```

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

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

```
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: 'Unicram',
    loves: [ 'energon' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
]
```

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

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

```
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'
  }
]
```

2.3.2) 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: 0})
```

```
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
  }
]
```

2.3.3) Найти всех единорогов, не имеющих ключ vampires.

```
db.unicorns.find({vampires: {$exists: false}}, {_id: 0})
```

```
learn> db.unicorns.find({vampires: {$exists: false}}, {_id: 0})
[
  {
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

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

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: {$slice: 1}, _id: 0}).sort({name: 1})
[
  {
    name: 'Dunx',
    loves: [ 'grape' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 680,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Kenny',
    loves: [ 'grape' ],
    weight: 630,
    gender: 'm',
    vampires: 39
  },
  {
    name: 'Pilot',
    loves: [ 'apple' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    name: 'Raleigh',
    loves: [ 'apple' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    name: 'Rococoedles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    name: 'Unicorn',
    loves: [ 'cucumber' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

3.1.1) 1) Создайте коллекцию towns, включающую следующие документы. 2) Сформировать запрос, который возвращает список городов с независимыми мэрами (party="I") . Вывести только название города и информацию о мэре. 3) Сформировать запрос, который возвращает список беспартийных мэров (party отсутствует) . Вывести только название города и информацию о мэре.

```
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"
  }
}
})

```

```

learn> db.towns.insertMany([
...   {
...     name: "Punxsutanney",
...     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('6831ccb4d9827134f142fff6'),
    '1': ObjectId('6831ccb4d9827134f142fff7'),
    '2': ObjectId('6831ccb4d9827134f142fff8')
  }
}

```


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

```
learn> db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 1, _id: 0})
[
  {
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
learn>
```

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

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

3.1.2) Сформировать функцию для вывода списка самцов единорогов. Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке. Вывести результат, используя forEach.

```
function printTopMaleUnicorns() {
  db.unicorns.find({gender: 'm'})
    .sort({name: 1})
    .limit(2)
    .forEach(doc => printjson(doc));
}
```

printTopMaleUnicorns()

```
learn> function printTopMaleUnicorns() {
...   db.unicorns.find({gender: 'm'})
...     .sort({name: 1})
...     .limit(2)
...     .forEach(doc => printjson(doc));
... }
...
[Function: printTopMaleUnicorns]
learn> printTopMaleUnicorns()
{
  _id: ObjectId('6831c380d9827134f142fff5'),
  name: 'Dunk',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('6831c364d9827134f142ffea'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

3.2.1) Вывести количество самок единорогов весом от полутонны до 600 кг.

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

```
learn> db.unicorns.find({gender: 'f', weight: { $gte: 500, $lte: 600 }}).count()
(node:30074) [MONGODB DRIVER] Warning: cursor.count is deprecated and will be removed in a future version. Use `cursor.countDocuments` instead
(Use `node --trace-warnings ...` to show where the warning was created)
2
```

3.2.2) Вывести список предпочтений.

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

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

3.2.3) Посчитать количество особей единорогов обоих полов.

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

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

3.3.1) Выполнить команду. Проверить содержимое коллекции unicorns.

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

Из-за более новой версии mongoDB не работает save, вместо него надо использовать insertOne

```
learn> db.unicorns.aggregate([{$group: {_id: "$gender", count: { $sum: 1 }}}])
[ { _id: 'm', count: 7 }, { _id: 'f', count: 5 } ]
learn> db.unicorns.save({name: 'Barney', loves: ['grape'],
... weight: 340, gender: 'm'})
TypeError: db.unicorns.save is not a function
learn> db.unicorns.save({name: 'Barney', loves: ['grape'], weight: 340, gender: 'm'})
TypeError: db.unicorns.save is not a function
learn>
```

3.3.2) Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вампира. Проверить содержимое коллекции unicorns.

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

```
db.unicorns.find({name: «Ayna»})
```

```
[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
}
[learn> db.unicorns.find({name: "Ayna"})
[
  {
    _id: ObjectId('6831c364d9827134f142ffef'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
learn>
```

3.3.3) Для самца единорога Raleigh внести изменения в БД: теперь он любит рэдбул. Проверить содержимое коллекции unicorns.

```
db.unicorns.update({ name: "Raleigh" }, { $addToSet: { loves: "redbull" } })
```

```
db.unicorns.find({name: "Raleigh"})
```

```
[learn> db.unicorns.update({ name: "Raleigh" }, { $addToSet: { loves: "redbull" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
[learn> db.unicorns.find({name: "Raleigh"})
[
  {
    _id: ObjectId('6831c364d9827134f142fff1'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
learn>
```

3.3.4) Всем самцам единорогов увеличить количество убитых вампиров на 5. Проверить содержимое коллекции `unicorns`.

```
db.unicorns.update({ gender: "m" }, { $inc:{ vampires: 5 } }, {multi:true})
```

```
db.unicorns.find( { gender: "m" } )
```

```

acknowledged: true,
insertedAt: null,
matchedCount: 7,
modifiedCount: 2,
upsertedCount: 0
},
name: 'uniques find( gender: "m" )'
}

{
  id: ObjectId('58212b4b95271247349f7f6c'),
  name: 'harry',
  team: [ 'harry', 'jacksal' ],
  weight: 630,
  gender: 'm',
  volume: 64
},
{
  id: ObjectId('58212b4b95271247349f7f6d'),
  name: 'lucaron',
  team: [ 'astron', 'redbull' ],
  weight: 990,
  gender: 'm',
  volume: 107
},
{
  id: ObjectId('58212b4b95271247349f7f6e'),
  name: 'hasekaceline',
  team: [ 'apple' ],
  weight: 570,
  gender: 'w',
  volume: 388
},
{
  id: ObjectId('58212b4b95271247349f7f70'),
  name: 'harry',
  team: [ 'apple', 'team 1' ],
  weight: 640,
  gender: 'm',
  volume: 64
},
{
  id: ObjectId('58212b4b95271247349f7f71'),
  name: 'haseka',
  team: [ 'apple', 'ugr', 'redbull' ],
  weight: 671,
  gender: 'w',
  volume: 7
},
{
  id: ObjectId('58212b4b95271247349f7f72'),
  name: 'felic',
  team: [ 'apple', 'actonshar' ],
  weight: 620,
  gender: 'w',
  volume: 59
},
{
  id: ObjectId('58212b4b95271247349f7f73'),
  name: 'lucan',
  team: [ 'apple', 'actonshar' ],
  weight: 784,
  gender: 'w',
  volume: 179
}
}
}

```

3.3.5) Изменить информацию о городе Портланд: мэр этого города теперь беспартийный. Проверить содержимое коллекции towns.

```
db.towns.update( { name: "Portland" }, { $unset: { "mayor.party": "" } })
```

```
db.towns.find({ name: "Portland" })
```

```
learn> db.towns.update({ name: "Portland" }, { $unset: { "mayor.party": "" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({ name: "Portland" })
[
  {
    _id: ObjectId('6831ccb4d9827134f142fff8'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

3.3.6) Изменить информацию о самце единорога Pilot: теперь он любит и шоколад. Проверить содержимое коллекции unicorns.

```
db.unicorns.update({ name: "Pilot" }, { $addToSet: { loves: "chocolate" } })
```

```
db.unicorns.find({ name: "Pilot" })
```

```
learn> db.unicorns.update({ name: "Pilot" }, { $addToSet: { loves: "chocolate" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Pilot" })
[
  {
    _id: ObjectId('6831c364d9827134f142fff3'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
learn> 
```

3.3.7) Изменить информацию о самке единорога Aurora: теперь она любит еще и сахар, и лимоны. Проверить содержимое коллекции unicorns.

```
db.unicorns.updateOne({ name: "Aurora" }, { $addToSet: { loves: { $each: ["sugar", "lemon"]} } })
```

```
db.unicorns.find({ name: "Aurora" })
```

```
learn> db.unicorns.update({ 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('6831c364e9827134f142ffeb'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

3.4.1) Создайте коллекцию towns, включающую следующие документы. Удалите документы с беспартийными мэрами. Проверьте содержание коллекции. Очистите коллекцию. Просмотрите список доступных коллекций.

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

```
db.towns.insertMany([
  {
    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"
    }
  }
])
```

```

    }
  }
})

```

```

[learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 3 }
learn> db.towns.insertMany([
...   {
...     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"
...     }
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68336ef8d9827134f142fff9'),
    '1': ObjectId('68336ef8d9827134f142fffa'),
    '2': ObjectId('68336ef8d9827134f142fffb')
  }
}

```

db.towns.deleteMany({ "mayor.party": { \$exists: false } })

db.towns.find()

db.towns.deleteMany({})

show collections

```
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 2 }
learn> show collections
[
  towns
]
learn>
{
  _id: ObjectId('68336ef8d9827134f142ffffb'),
  name: 'Portland',
  population: 528000,
  last_census: ISODate('2009-07-20T00:00:00.000Z'),
  famous_for: [ 'beer', 'food' ],
  mayor: { name: 'Sam Adams', party: 'D' }
}
learn>
```

4.1.1) Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание. Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания. Проверьте содержание коллекции единорогов.

```
db.habitats.insertMany([
  {
    _id: "sky",
    name: "Sky Haven",
    description: "Floating islands high above the clouds where unicorns soar freely."
  },
  {
    _id: "meadow",
    name: "Blossom Meadow",
    description: "Endless flower fields with soft grass and gentle breezes."
  },
  {
    _id: "ice",
    name: "Frozen Valley",
    description: "Glacial terrain where snowflakes shimmer and silence reigns."
  }
])
```

```
db.unicorns.updateOne({ name: "Ayna" }, { $set: { habitat: { $ref: "habitats", $id: «ice» } } })
```

```
db.unicorns.updateOne({ name: "Nimue" }, { $set: { habitat: { $ref: "habitats", $id: «meadow» } } })
```



```
db.unicorns.updateOne({ name: "Leia" }, { $set: { habitat: { $ref: "habitats", $id: «sky» } } })
```

```
db.unicorns.find()
```

```
learn> db.habitats.insertMany([
...   {
...     _id: "sky",
...     name: "Sky Haven",
...     description: "Floating islands high above the clouds where unicorns soar freely."
...   },
...   {
...     _id: "meadow",
...     name: "Blossom Meadow",
...     description: "Endless flower fields with soft grass and gentle breezes."
...   },
...   {
...     _id: "ice",
...     name: "Frozen Valley",
...     description: "Glacial terrain where snowflakes shimmer and silence reigns."
...   }
... ])
{
  acknowledged: true,
  insertedIds: { '0': 'sky', '1': 'meadow', '2': 'ice' }
}
learn> show collections
habitats
towns
unicorns
learn> █
```

```
{
  _id: ObjectId('6831c364d9827134f142ffef'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 800,
  gender: 'f',
  vampires: 51,
  habitat: DBRef('habitats', 'ice')
},
{
```

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

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

```

learn> db.unicorns.insertMany([
...   { name: 'Hurry', dob: new Date(1992, 2, 13, 7, 47), loves: ['carrot', 'papaya'], weight: 600, gender: 'm', vampires: 63 },
...   { name: 'Aurora', dob: new Date(1991, 0, 24, 13, 0), loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43 },
...   { name: 'Unicorn', dob: new Date(1973, 1, 9, 21, 14), loves: ['energon', 'redbull'], weight: 584, gender: 'm', vampires: 182 },
...   { name: 'Koooodies', dob: new Date(1979, 7, 18, 18, 44), loves: ['apple'], weight: 373, gender: 'm', vampires: 99 },
...   { name: 'Solnara', dob: new Date(1985, 6, 4, 2, 1), loves: ['apple', 'carrot', 'chocolate'], weight: 530, gender: 'f', vampires: 40 },
...   { name: 'Ayra', dob: new Date(1978, 4, 7, 8, 30), loves: ['strawberry', 'lemon'], weight: 753, gender: 'f', vampires: 40 },
...   { name: 'Kerry', dob: new Date(1997, 5, 1, 10, 42), loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39 },
...   { name: 'Raleigh', dob: new Date(2005, 4, 3, 0, 37), loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2 },
...   { name: 'Leila', dob: new Date(2001, 9, 8, 14, 53), loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33 },
...   { name: 'Pilot', dob: new Date(1957, 2, 1, 5, 3), loves: ['apple', 'watermelon'], weight: 550, gender: 'm', vampires: 54 },
...   { name: 'Nimue', dob: new Date(1959, 11, 29, 16, 15), loves: ['grape', 'carrot'], weight: 540, gender: 'f' },
...   { name: 'Dunx', dob: new Date(1576, 6, 18, 18, 14), loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 155 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6813759549d27134f142466f'),
    '1': ObjectId('6813759549d27134f142466d'),
    '2': ObjectId('6813759549d27134f142466e'),
    '3': ObjectId('6813759549d27134f142466f'),
    '4': ObjectId('6813759549d27134f1430000'),
    '5': ObjectId('6813759549d27134f1430001'),
    '6': ObjectId('6813759549d27134f1430002'),
    '7': ObjectId('6813759549d27134f1430003'),
    '8': ObjectId('6813759549d27134f1430004'),
    '9': ObjectId('6813759549d27134f1430005'),
    '10': ObjectId('6813759549d27134f1430006'),
    '11': ObjectId('6813759549d27134f1430007')
  }
}
learn> db.unicorns.createIndex({ name: 1 }, { unique: true })
name_1

```

4.3.1) Получите информацию о всех индексах коллекции `unicorns`. Удалите все индексы, кроме индекса для идентификатора. Попробуйте удалить индекс для идентификатора.

`db.unicorns.getIndexes()`

```

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()`

```

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>

```

`db.unicorns.dropIndex(<<_id>>)`

```

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

```

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

db.numbers.find().sort({ \$natural: -1 }).limit(4).explain("executionStats")

```
learn> db.numbers.find().sort({ $natural: -1 }).limit(4).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: '0F2303C1',
    planCacheShapeHash: 'BF2303EE',
    planCacheKey: '7DF350EE',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansExplainedReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: { stage: 'COLLSCAN', direction: 'backward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
      isCached: false,
      stage: 'LIMIT',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 3,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      limitAmount: 4,
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 4,
        executionTimeMillisEstimate: 0,
        works: 4,
        advanced: 4,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 0,
        direction: 'backward',
        docsExamined: 4
      }
    }
  },
  queryShapeHash: '71C04F1B0002B31344A2B08A520E31838D2B0F418D60563B2C78061847B220E4',
  command: {
    find: 'numbers',
    filter: {},
    sort: { '$natural': -1 },
    limit: 4,
    '$db': 'learn'
  },
  serverInfo: {
    host: 'MacBook-Pro-kovalev.local',
    port: 27027,
    version: '3.0.3',
    gitVersion: 'f182ef116d531ee4bb593143e4c554f4e90e415'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeInMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForKegex: 1
  },
  ok: 1
}
learn>
```

```
db.numbers.getIndexes()
```

```
db.numbers.find().sort({ $natural: -1 }).limit(4).explain(«executionStats")
```

[illegible]

Анализ:

- Первый запрос без индекса: stage: 'COLLSCAN', executionTimeMillis: 0
- Второй запрос с индексом: stage: 'COLLSCAN', executionTimeMillis: 1

Оба запроса не используют индексы из-за

MongoDB просто прочитал документы напрямую в порядке их физического расположения (по \$natural), начиная с конца ('backward') в обоих случаях. Использование \$natural игнорирует любые индексы. Поэтому даже создать индекс по value, он не будет задействован при сортировке по \$natural. Из-за этого два запроса по сути одинаковы, чтобы использовать индекс нужно сделать другой запрос, например: `db.numbers.find().sort({ value: -1 }).limit(4).explain(«executionStats»)`.

В данном случае индекс будет работать быстрее.

Выводы по лабораторной работе:

Лабораторная работа позволила получить комплексное представление о работе с MongoDB: от базовых операций до использования индексов, связей между коллекциями и анализа производительности запросов. Эти знания являются основой для разработки современных, гибких и масштабируемых приложений на базе NoSQL.