Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной

техники Кафедра вычислительных систем

ОТЧЕТ
по лабораторной работе № 6 на тему:
«Работа с БД в СУБД MongoDB**»**

Выполнил: Кисель Софья Михайловна
Группа: К3240

Проверил:

Санкт-Петербург
2025

Цель работы

Овладеть практическими навыками работы с CRUD-операциями, вложенными объектами, агрегацией, изменением данных, ссылками и индексами в MongoDB.

Часть 1 - установка

С помощью brew установила mongo были небольшие проблемы с версией поэтому пришлось несколько раз переустанавливать но все же увидела долгожданный тест и все успешно установилось

```
v18.20.8
sonic@MacBook-Pro-Sonic ~ % brew uninstall --force mongosh

sonic@MacBook-Pro-Sonic ~ % npm install -g mongosh@1.8
npm warn deprecated acorn-numeric-separator@0.3.6: acorn>=7.4 supports numeric separators

added 349 packages in 49s

38 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.4.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.4.2
npm notice To update run: npm install -g npm@11.4.2
npm notice
sonic@MacBook-Pro-Sonic ~ % mongosh --version
1.8.2
sonic@MacBook-Pro-Sonic ~ % brew services start mongodb-community@6.0
Service `mongodb-community@6.0` already started, use `brew services restart mongodb-community@6.0` to restart.
sonic@MacBook-Pro-Sonic ~ % mongosh
Current Mongosh Log ID: 6858ee41805987d147d15eb3
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.8.2
Using MongoDB:          6.0.24
Using Mongosh:          1.8.2

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.
```

С которой попытки но пофиксилось и тест ура

```
test>

test> use learn
switched to db learn
learn> db.unicorns.insertMany([
```

Практическое задание 2.1.1

Создание базы данных и коллекции

```
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("6858f091805987d147d15eb4"),
    '1': ObjectId("6858f091805987d147d15eb5"),
    '2': ObjectId("6858f091805987d147d15eb6"),
    '3': ObjectId("6858f091805987d147d15eb7"),
    '4': ObjectId("6858f091805987d147d15eb8"),
    '5': ObjectId("6858f091805987d147d15eb9"),
    '6': ObjectId("6858f091805987d147d15eba"),
    '7': ObjectId("6858f091805987d147d15ebb"),
    '8': ObjectId("6858f091805987d147d15ebc"),
    '9': ObjectId("6858f091805987d147d15ebd"),
    '10': ObjectId("6858f091805987d147d15ebe")
  }
}
learn> db.unicorns.insertOne({
...   name: 'Dunx',
...   loves: ['grape', 'watermelon'],
...   weight: 704,
...   gender: 'm',
...   vampires: 165
... })
{
  acknowledged: true,
  insertedId: ObjectId("6858f094805987d147d15ebf")
}
learn> db.unicorns.find().pretty()
[
  {
    _id: ObjectId("6858f091805987d147d15eb4"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("6858f091805987d147d15eb5"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
```

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6858f091805987d147d15eb4"),
    '1': ObjectId("6858f091805987d147d15eb5"),
    '2': ObjectId("6858f091805987d147d15eb6"),
    '3': ObjectId("6858f091805987d147d15eb7"),
    '4': ObjectId("6858f091805987d147d15eb8"),
    '5': ObjectId("6858f091805987d147d15eb9"),
    '6': ObjectId("6858f091805987d147d15eba"),
    '7': ObjectId("6858f091805987d147d15ebb"),
    '8': ObjectId("6858f091805987d147d15ebc"),
    '9': ObjectId("6858f091805987d147d15ebd"),
    '10': ObjectId("6858f091805987d147d15ebe")
  }
}
```

Практическое задание 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.

```
learn> db.unicorns.countDocuments()
12
learn> db.unicorns.find({loves: 'apple'}).pretty()
[
  {
    _id: ObjectId("6858f091805987d147d15eb7"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("6858f091805987d147d15eb8"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("6858f091805987d147d15ebb"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId("6858f091805987d147d15ebc"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId("6858f091805987d147d15ebd"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
]
learn> db.unicorns.findOne()
{
  _id: ObjectId("6858f091805987d147d15eb4"),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
learn> db.unicorns.find({gender: 'm'}).sort({name: 1}).pretty()
[
  {
    _id: ObjectId("6858f094805987d147d15ebf"),
    name: 'Dunx',
```

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1}).pretty()
[
  {
    _id: ObjectId("6858f094805987d147d15ebf"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("6858f091805987d147d15eb4"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("6858f091805987d147d15eba"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId("6858f091805987d147d15ebd"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId("6858f091805987d147d15ebb"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId("6858f091805987d147d15eb7"),
```

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3).pretty()
[
  {
    _id: ObjectId("6858f091805987d147d15eb5"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("6858f091805987d147d15eb9"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId("6858f091805987d147d15ebc"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn> db.unicorns.find({
...   gender: 'f',
```

Практическое задание 2.2.2

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

```
  { name: 'Nimue', loves: [ 'grape', 'carrot' ], gender: 'f' }
]
learn> db.unicorns.find({gender: 'm'}).sort({name: 1}).pretty()
[
  {
    _id: ObjectId("6858f094805987d147d15ebf"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("6858f091805987d147d15eb4"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("6858f091805987d147d15eba"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId("6858f091805987d147d15ebd"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
```

Практическое задание 2.2.3

Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({_id: -1}).toArray().forEach(unicorn => {
...     printjson({
...         name: unicorn.name,
...         gender: unicorn.gender,
...         loves: unicorn.loves,
...         weight: unicorn.weight,
...         vampires: unicorn.vampires || 0
...     });
... });
{
  name: 'Raleigh',
  gender: 'm',
  loves: [
    'redbull'
  ],
  weight: 500,
  vampires: 10
}
{
  name: 'Barny',
  gender: 'm',
  loves: [
    'apple',
    'carrot'
  ],
  weight: 520,
  vampires: 12
}
{
  name: 'Dunx',
  gender: 'm',
  loves: [
    'grape',
    'watermelon',
    'marshmallow'
  ],
  weight: 704,
  vampires: 175
}
{
  name: 'Nimue',
  gender: 'f',
  loves: [
    'grape',
    'carrot',
    'coconut',
    'marshmallow'
  ],
```

Практическое задание 2.2.4

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

```
learn> db.unicorns.find(
...   {},
...   {
...     name: 1,
...     loves: { $slice: -1 },
...   _id: 0
...   }
... ).pretty()
[
  { name: 'Horny', loves: [ 'marshmallow' ] },
  { name: 'Unicrom', loves: [ 'marshmallow' ] },
  { name: 'Roooooodles', loves: [ 'marshmallow' ] },
  { name: 'Solnara', loves: [ 'marshmallow' ] },
  { name: 'Ayna', loves: [ 'marshmallow' ] },
  { name: 'Pilot', loves: [ 'marshmallow' ] },
  { name: 'Nimue', loves: [ 'marshmallow' ] },
  { name: 'Dunx', loves: [ 'marshmallow' ] },
  { name: 'Nimue', loves: [ 'marshmallow' ] }
]
learn> db.unicorns.find(
...   {
...     vampires: { $exists: false }
...   },
...   {
...     name: 1,
...     loves: { $slice: -1 },
...     _id: 0
...   }
... ).pretty()
[
  { name: 'Solnara', loves: [ 'marshmallow' ] },
  { name: 'Ayna', loves: [ 'marshmallow' ] },
  { name: 'Nimue', loves: [ 'marshmallow' ] },
  { name: 'Nimue', loves: [ 'marshmallow' ] }
]
learn>
```

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: 1,
...     gender: 1,
...     weight: 1,
...     loves: 1,
...     vampires: 1
...   }
... ).sort({ weight: -1 }).toArray().forEach(unicorn => printjson(unicorn));
{
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate',
    'coconut',
    'marshmallow'
  ],
  weight: 605,
  gender: 'f'
}
{
  name: 'Nimue',
  loves: [
    'grape',
    'carrot',
    'coconut',
    'marshmallow'
  ],
  weight: 540,
  gender: 'f'
}
```

Практическое задание 2.3.4

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

```
learn> db.unicorns.find(
...   {
...     vampires: { $exists: false }
...   },
...   {
...     name: 1,
...     loves: { $slice: -1 },
...     _id: 0
...   }
... ).pretty()
[
  { name: 'Solnara', loves: [ 'marshmallow' ] },
  { name: 'Ayna', loves: [ 'marshmallow' ] },
  { name: 'Nimue', loves: [ 'marshmallow' ] },
  { name: 'Nimue', loves: [ 'marshmallow' ] }
]
learn>

learn>

learn> db.unicorns.find(
...     {gender: 'm'},
...     {_id: 0, name: 1, firstLove: {$arrayElemAt: ['$loves', 0]}}
... ).sort({name: 1})
[
  { name: 'Dunx', firstLove: 'grape' },
  { name: 'Horny', firstLove: 'carrot' },
  { name: 'Pilot', firstLove: 'apple' },
  { name: 'Roooooodles', firstLove: 'apple' },
  { name: 'Unicrom', firstLove: 'energon' }
]
learn>

learn>
```

Практическое задание 3.1.1

Создайте коллекцию towns и выполните выборки по мэрам с party="I" и без party.

```
learn>

learn> db.towns.insertMany([
...     {
...         name: "Springfield",
...         mayor: {
...             name: "Joe Quimby",
...             party: "I",
...             yearsInOffice: 8
...         },
...         population: 30000
...     },
...     {
...         name: "Shelbyville",
...         mayor: {
...             name: "Arnold Pops",
...             party: "D",
...             yearsInOffice: 4
...         },
...         population: 28000
...     },
...     {
...         name: "Ogdenville",
...         mayor: {
...             name: "Ruth Meyer",
...             party: "I",
...             yearsInOffice: 12
...         },
...         population: 25000
...     },
...     {
...         name: "North Haverbrook",
...         mayor: {
...             name: "Sam Clover",
...             yearsInOffice: 2
...         },
...         population: 22000
...     },
...     {
...         name: "Capital City",
...         mayor: {
...             name: "Diamond Joe",
...             party: "R",
...             yearsInOffice: 6
...         },
...         population: 500000
...     }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("68590520805987d147d15ec1"),
    '1': ObjectId("68590520805987d147d15ec2"),
```

```
learn> db.towns.find({
...     "mayor.party": "I"
... }, {
...     name: 1,
...     "mayor.name": 1,
...     "mayor.yearsInOffice": 1,
...     _id: 0
... })
[
  {
    name: 'Springfield',
    mayor: { name: 'Joe Quimby', yearsInOffice: 8 }
  },
  {
    name: 'Ogdenville',
    mayor: { name: 'Ruth Meyer', yearsInOffice: 12 }
  }
]
learn> db.towns.find({
...     "mayor.party": { $exists: false }
... }, {
...     name: 1,
...     "mayor.name": 1,
...     "mayor.yearsInOffice": 1,
...     _id: 0
... })
[
  {
    name: 'North Haverbrook',
    mayor: { name: 'Sam Clover', yearsInOffice: 2 }
  }
]
learn> db.towns.find({
...     $or: [
...         { "mayor.party": { $ne: "I" } },
...         { "mayor.party": { $exists: false } }
...     ]
... }, {
...     name: 1,
...     "mayor.name": 1,
...     "mayor.party": 1,
...     _id: 0
... })
[
  { name: 'Shelbyville', mayor: { name: 'Arnold Pops', party: 'D' } },
  { name: 'North Haverbrook', mayor: { name: 'Sam Clover' } },
  { name: 'Capital City', mayor: { name: 'Diamond Joe', party: 'R' } }
]
learn>

learn>

learn> function findMaleUnicorns(limit = 2) {
...     return db.unicorns.find(
...         { gender: 'm' },
...         { _id: 0, name: 1, loves: 1 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("68590520805987d147d15ec1"),
    '1': ObjectId("68590520805987d147d15ec2"),
    '2': ObjectId("68590520805987d147d15ec3"),
    '3': ObjectId("68590520805987d147d15ec4"),
    '4': ObjectId("68590520805987d147d15ec5")
  }
}
learn> db.towns.find({
...     "mayor.party": "I"
... }, {
...     name: 1,
...     "mayor.name": 1,
...     "mayor.yearsInOffice": 1,
...     _id: 0
... })
[
  {
    name: 'Springfield',
    mayor: { name: 'Joe Quimby', yearsInOffice: 8 }
  },
  {
    name: 'Ogdenville',
    mayor: { name: 'Ruth Meyer', yearsInOffice: 12 }
  }
]
learn> db.towns.find({
...     "mayor.party": { $exists: false }
... }, {
...     name: 1,
...     "mayor.name": 1,
...     "mayor.yearsInOffice": 1,
...     _id: 0
... })
[
  {
    name: 'North Haverbrook',
    mayor: { name: 'Sam Clover', yearsInOffice: 2 }
  }
]
```

Практическое задание 3.1.2

Сформировать функцию для вывода списка самцов единорогов и вывести первых двух.

```
learn>

learn> function findMaleUnicorns(limit = 2) {
...     return db.unicorns.find(
...         { gender: 'm' },
...         { _id: 0, name: 1, loves: 1 }
...     )
...     .sort({ name: 1 })
...     .limit(limit)
...     .toArray();
... }
[Function: findMaleUnicorns]
learn>

learn>

learn> findMaleUnicorns();
[
  { name: 'Dunx', loves: [ 'grape', 'watermelon', 'marshmallow' ] },
  { name: 'Horny', loves: [ 'carrot', 'papaya', 'marshmallow' ] }
]
learn> function findMaleUnicornsWithFirstLove(limit = 2) {
...     return db.unicorns.aggregate([
...         { $match: { gender: 'm' } },
...         { $project: {
...             _id: 0,
...             name: 1,
...             firstLove: { $arrayElemAt: ['$loves', 0] }
...         }},
...         { $sort: { name: 1 } },
...         { $limit: limit }
...     ]).toArray();
... }
[Function: findMaleUnicornsWithFirstLove]
learn>

learn>

learn> findMaleUnicornsWithFirstLove();
[
  { name: 'Dunx', firstLove: 'grape' },
  { name: 'Horny', firstLove: 'carrot' }
]
learn>
```

Практическое задание 3.2.1

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
learn> db.unicorns.countDocuments({
...     gender: 'f',
...     weight: { $gte: 500, $lte: 600 }
... })
2
learn>

learn>

learn> db.unicorns.aggregate([
...   { $unwind: "$loves" },
...   { $group: { _id: "$loves" } },
...   { $project: { _id: 0, preference: "$_id" } },
...   { $sort: { preference: 1 } }
... ])
[
  { preference: 'apple' },
  { preference: 'carrot' },
  { preference: 'chocolate' },
  { preference: 'coconut' },
  { preference: 'energon' },
  { preference: 'grape' },
  { preference: 'lemon' },
  { preference: 'marshmallow' },
  { preference: 'papaya' },
  { preference: 'redbull' },
  { preference: 'strawberry' },
  { preference: 'watermelon' }
]
```

И предпочтения тоже тут

Практическое задание 3.2.2

Вывести список предпочтений.

Практическое задание 3.2.3

Посчитать количество особей единорогов обоих полов.

```
learn> db.unicorns.aggregate([
...   {
...     $group: {
...       _id: "$gender",
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       gender: "$_id",
...       count: 1
...     }
...   }
... ])
[ { count: 4, gender: 'f' }, { count: 5, gender: 'm' } ]
learn>

learn>

learn> db.unicorns.insertOne({
...     name: "Barny",
...     gender: "m",
...     loves: ["apple", "carrot"],
...     weight: 520,
...     vampires: 12
... })
{
  acknowledged: true,
  insertedId: ObjectId("685907c3805987d147d15ec6")
}
```

Практическое задание 3.3.1

Добавить самца

```
learn> db.unicorns.insertOne({
...     name: "Barny",
...     gender: "m",
...     loves: ["apple", "carrot"],
...     weight: 520,
...     vampires: 12
... })
{
  acknowledged: true,
  insertedId: ObjectId("685907c3805987d147d15ec6")
}
learn>
```

Barny

Практическое задание 3.3.2

Обновить Ayna: вес 800, вампиры 51.

```
learn> db.unicorns.findOne({ name: "Ayna" }, { _id: 0 })
{
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon', 'coconut', 'marshmallow' ],
  weight: 733,
  gender: 'f'
}
learn> db.unicorns.updateOne(
...   { name: "Ayna" },
...   {
...     $set: {
...       weight: 800,
...       vampires: 51,
...       gender: "f",
...       loves: ["apple"]     }
...   },
...   { upsert: true }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Практическое задание 3.3.3

Обновить Raleigh: добавить redbull в loves.

```
learn> db.unicorns.updateOne(
...   { name: "Raleigh" },
...   {
...     $setOnInsert: {
...       name: "Raleigh",
...       gender: "m",
...       weight: 500,
...       vampires: 10,
...       loves: ["apple", "redbull"]
...     }
...   },
...   { upsert: true }
... )
{
  acknowledged: true,
  insertedId: ObjectId("68590961b5ceaf67634fa14d"),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
learn>

learn>

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

Практическое задание 3.3.4
Увеличить количество убитых вампиров у всех самцов на 5.

```
learn> db.unicorns.find(
...   { gender: 'm' },
...   { _id: 0, name: 1, vampires: 1 }
... ).sort({ name: 1 })
[
  { name: 'Barny', vampires: 12 },
  { name: 'Dunx', vampires: 175 },
  { name: 'Horny', vampires: 63 },
  { name: 'Pilot', vampires: 64 },
  { name: 'Raleigh', vampires: 10 },
  { name: 'Roooooodles', vampires: 99 },
  { name: 'Unicrom', vampires: 192 }
]
learn>
```

Практическое задание 3.3.5
Убрать партию у мэра Портланда

```
learn>  db.towns.updateOne(
...     { name: 'Portland' },
...     { $unset: { 'mayor.party': "" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
learn> db.unicorns.updateOne(
...   { name: 'Pilot' },
...   { $addToSet: { loves: 'chocolate' } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.findOne(
...   { name: 'Pilot' },
...   { _id: 0, name: 1, loves: 1 }
... )
{
  name: 'Pilot',
  loves: [ 'apple', 'watermelon', 'marshmallow', 'chocolate' ]
}
learn>
```

И тут добавили
Практическое задание 3.3.6
Обновить Pilot: добавить chocolate в loves.

Практическое задание 3.3.7
Обновить Aurora: добавить sugar и lemon в loves.

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

Практическое задание 3.4.1
Удалить беспартийных мэров, очистить коллекцию, просмотреть коллекции.

```
learn> db.towns.deleteMany({
...   "mayor.party": { $exists: false }
... })
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 4 }
learn> show collections
towns
unicorns
```

Практическое задание 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.

```
learn> db.habitats.insertMany([
...   {
...     _id: "forest",
...     fullName: "Волшебный лес",
...     description: "Густой лес с древними деревьями, где единороги прячутся среди зарослей"
...   },
...   {
...     _id: "mountains",
...     fullName: "Хрустальные горы",
...     description: "Высокие горы с чистейшими ледниковыми озерами"
...   },
...   {
...     _id: "meadow",
...     fullName: "Радужные луга",
...     description: "Бескрайние цветущие поля под постоянной радугой"
...   }
... ])
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'mountains', '2': 'meadow' }
}
learn> db.unicorns.updateMany(
...   { name: { $in: ["Aurora", "Pilot"] } },
...   { $set: { habitatId: "forest" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>

learn> db.unicorns.updateOne(
...   { name: "Nimue" },
...   { $set: { habitatId: "meadow" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
learn> db.unicorns.updateOne(
...   { name: "Unicrom" },
...   { $set: { habitatId: "mountains" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.aggregate([
...   {
...     $lookup: {
...       from: "habitats",
...       localField: "habitatId",
...       foreignField: "_id",
...       as: "habitatInfo"
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       name: 1,
...       gender: 1,
...       habitat: { $arrayElemAt: ["$habitatInfo", 0] }
...     }
...   }
... ])
[
  { name: 'Horny', gender: 'm' },
  {
    name: 'Unicrom',
    gender: 'm',
    habitat: {
      _id: 'mountains',
      fullName: 'Хрустальные горы',
      description: 'Высокие горы с чистейшими ледниковыми озерами'
    }
  },
  { name: 'Roooooodles', gender: 'm' },
  { name: 'Solnara', gender: 'f' },
  { name: 'Ayna', gender: 'f' },
  {
    name: 'Pilot',
    gender: 'm',
    habitat: {
      _id: 'forest',
      fullName: 'Волшебный лес',
      description: 'Густой лес с древними деревьями, где единороги прячутся среди зарослей'
```

Практическое задание 4.2.1

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

Ошибка ибо был обноружен дубликат

```
learn> try {
...   db.unicorns.createIndex(
...     { name: 1 },
...     {
...       unique: true,
...       name: "unique_name_index"
...     }
...   )
... } catch (e) {
...   print("Ошибка создания индекса: " + e.message)
... }
Ошибка создания индекса: Index build failed: f3fedd28-3151-45a2-b563-23b2b
corns index: unique_name_index dup key: { name: "Nimue" }

learn> try {
...   db.unicorns.createIndex(
...     { name: 1 },
...     {
...       unique: true,
...       name: "unique_name_index"
...     }
...   )
... } catch (e) {
...   print("Ошибка создания индекса: " + e.message)
... }
Ошибка создания индекса: Index build failed: 9cb99316-221d-4fe5-85b9-724c5
corns index: unique_name_index dup key: { name: "Nimue" }

learn>
```

```
learn>

learn> duplicates.forEach(function(group) {
...   print("Duplicate name: " + group._id);
...   group.docs.forEach(function(doc) {
...     printjson({_id: doc._id, name: doc.name});
...   });
... });
Duplicate name: Nimue
{
  _id: ObjectId("6858f091805987d147d15ebe"),
  name: 'Nimue'
}
{
  _id: ObjectId("6858f8e1805987d147d15ec0"),
  name: 'Nimue'
}

learn> duplicates.forEach(function(group) {
...   var idsToKeep = group.docs[0]._id;   db.unicorns.deleteMany({
...     name: group._id,
...     _id: { $ne: idsToKeep }
...   });
... });

learn> duplicates.forEach(function(group) {
...   var idsToKeep = group.docs[0]._id;   db.unicorns.deleteMany({
...     name: group._id,
...     _id: { $ne: idsToKeep }
...   });
... });

learn> db.unicorns.createIndex(
...   { name: 1 },
...   {
...     unique: true,
...     name: "unique_name_index"
...   }
... );
unique_name_index
learn> try {
...   db.unicorns.insertOne({ name: "Nimue" });
...   print("Успешно добавлен Nimue");
...   db.unicorns.insertOne({ name: "Nimue" });
... } catch (e) {
...   print("Ошибка дубликата (как и ожидалось): " + e.message);
... }
Ошибка дубликата (как и ожидалось): E11000 duplicate key error collection: learn.unicor

learn> var duplicates = db.unicorns.aggregate([
...   {
```

Практическое задание 4.3.1

Получите информацию о всех индексах коллекции unicorns. Удалите все индексы, кроме индекса для идентификатора. Попытайтесь удалить индекс для идентификатора.

```
learn> indexes.forEach(function(index) {
...   if (index.name !== "_id_") {
...     db.unicorns.dropIndex(index.name);
...     print("Удален индекс: " + index.name);
...   }
... });
Удален индекс: weight_1
Удален индекс: unique_name_index

learn> try {
...   db.unicorns.dropIndex("_id_");
...   print("Индекс _id_ успешно удален");
... } catch (e) {
...   print("Ошибка при удалении индекса _id_: " + e.message);
... }
Ошибка при удалении индекса _id_: cannot drop _id index

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

learn>
```

Практическое задание 4.4.1

Создайте объемную коллекцию numbers, задействовав курсор. Выберите последние четыре документа. Проанализируйте план выполнения запроса. Сколько потребовалось времени на выполнение запроса? Создайте индекс для ключа value. Получите информацию о всех индексах коллекции numbers. Выполните запрос 2. Проанализируйте план выполнения запроса с установленным индексом. Сравните время выполнения запросов с индексом и без.

```
learn> print("Время выполнения: " + (endTime - startTime) + " мс");
Время выполнения: 34 мс

learn>

learn> var explain = db.numbers.find().sort({_id: -1}).limit(4).explain("executionStats");

learn> print("План выполнения:");
План выполнения:

learn> printjson({
...   executionTimeMillis: explain.executionStats.executionTimeMillis,
...   totalDocsExamined: explain.executionStats.totalDocsExamined,
...   executionStages: explain.executionStats.executionStages
... });
{
  executionTimeMillis: 0,
  totalDocsExamined: 4,
```

```
learn> for (let i = 1; i <= 10000; i++) {
...   bulk.insert({
...     value: i,
...     isEven: i % 2 === 0,
...     squared: i * i,
...     timestamp: new Date()
...   });
... }
{ nInsertOps: 10000, nUpdateOps: 0, nRemoveOps: 0, nBatches: 11 }
learn> bulk.execute();
{
  acknowledged: true,
  insertedCount: 10000,
  insertedIds: [
    { index: 0, _id: ObjectId("68592acc805987d147d185d9") },
    { index: 1, _id: ObjectId("68592acc805987d147d185da") },
    { index: 2, _id: ObjectId("68592acc805987d147d185db") },
    { index: 3, _id: ObjectId("68592acc805987d147d185dc") },
    { index: 4, _id: ObjectId("68592acc805987d147d185dd") },
    { index: 5, _id: ObjectId("68592acc805987d147d185de") },
    { index: 6, _id: ObjectId("68592acc805987d147d185df") },
    { index: 7, _id: ObjectId("68592acc805987d147d185e0") },
    { index: 8, _id: ObjectId("68592acc805987d147d185e1") },
    { index: 9, _id: ObjectId("68592acc805987d147d185e2") },
    { index: 10, _id: ObjectId("68592acc805987d147d185e3") },
    { index: 11, _id: ObjectId("68592acc805987d147d185e4") },
    { index: 12, _id: ObjectId("68592acc805987d147d185e5") },
    { index: 13, _id: ObjectId("68592acc805987d147d185e6") },
    { index: 14, _id: ObjectId("68592acc805987d147d185e7") },
    { index: 15, _id: ObjectId("68592acc805987d147d185e8") },
    { index: 16, _id: ObjectId("68592acc805987d147d185e9") },
    { index: 17, _id: ObjectId("68592acc805987d147d185ea") },
    { index: 18, _id: ObjectId("68592acc805987d147d185eb") },
    { index: 19, _id: ObjectId("68592acc805987d147d185ec") },
    { index: 20, _id: ObjectId("68592acc805987d147d185ed") },
    { index: 21, _id: ObjectId("68592acc805987d147d185ee") },
    { index: 22, _id: ObjectId("68592acc805987d147d185ef") },
    { index: 23, _id: ObjectId("68592acc805987d147d185f0") },
    { index: 24, _id: ObjectId("68592acc805987d147d185f1") },
    { index: 25, _id: ObjectId("68592acc805987d147d185f2") },
    { index: 26, _id: ObjectId("68592acc805987d147d185f3") },
    { index: 27, _id: ObjectId("68592acc805987d147d185f4") },
    { index: 28, _id: ObjectId("68592acc805987d147d185f5") },
    { index: 29, _id: ObjectId("68592acc805987d147d185f6") },
```

```
learn> printjson({
...   executionTimeMillis: explainWithoutIndex.executionStats.executionTimeMillis,
...   totalDocsExamined: explainWithoutIndex.executionStats.totalDocsExamined
... });
{
  executionTimeMillis: 19,
  totalDocsExamined: 20000
}

learn>

learn> print("\nСравнение производительности:");

Сравнение производительности:

learn> print("С индексом: " + timeWithIndex + " мс");
С индексом: 45 мс

learn> print("Без индекса: " + timeWithoutIndex + " мс");
Без индекса: 81 мс

learn> print("Разница: " + (timeWithoutIndex - timeWithIndex) + " мс");
Разница: 36 мс

learn> print("Ускорение: " + (timeWithoutIndex/timeWithIndex).toFixed(1) + "x");
Ускорение: 1.8x
```

Выполнив запросы видно что с индексом выполняется в два раза быстрее, но работает если для перебора

Вывод

В ходе выполнения этого задания я как студентка-разработчица освоила ключевые аспекты работы с MongoDB. На практике я научилась:

1. Создавать и наполнять коллекции данными, включая работу с объемными наборами
2. Формировать сложные запросы с фильтрацией, сортировкой и ограничением выборки
3. Анализировать производительность запросов через explain()
4. Создавать и использовать индексы для оптимизации поиска

Особенно ценным для меня стало понимание, как правильно проектировать структуру данных и индексы под конкретные запросы. В процессе возникли некоторые сложности - сначала не сразу получилось правильно составить агрегационные запросы и интерпретировать результаты explain, но методом проб и ошибок я разобралась с этими моментами.

А еще при установке бесконечные ошибки дали мне еще один раз понять, что обновлять все нужно во время и триста раз перепроверить версии все