Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет прикладной информатики
Образовательная программа Мобильные и сетевые технологии
Направление подготовки 09.03.03 Прикладная информатика

О Т Ч Е Т
О практической работе №6 по дисциплине «Проектирование и реализация баз данных»

Тема:
1. Введение в субд MongoDB. Установка MongoDB. Начало работы с бд.
2. Работа с БД в СУБД MongoDB.

Обучающийся: Михайлов Юрий Алексеевич, К3241

Преподаватель: Говорова Марина Михайловна

Санкт-Петербург,
2025

Цель работы

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

Ход работы (часть 1)

Как было показано в инструкции, устанавливаем MongoDB Server на ПК. Мне предложили установить последнюю 8 версию, и в папке bin я не нашел клиента и консоли mongo:

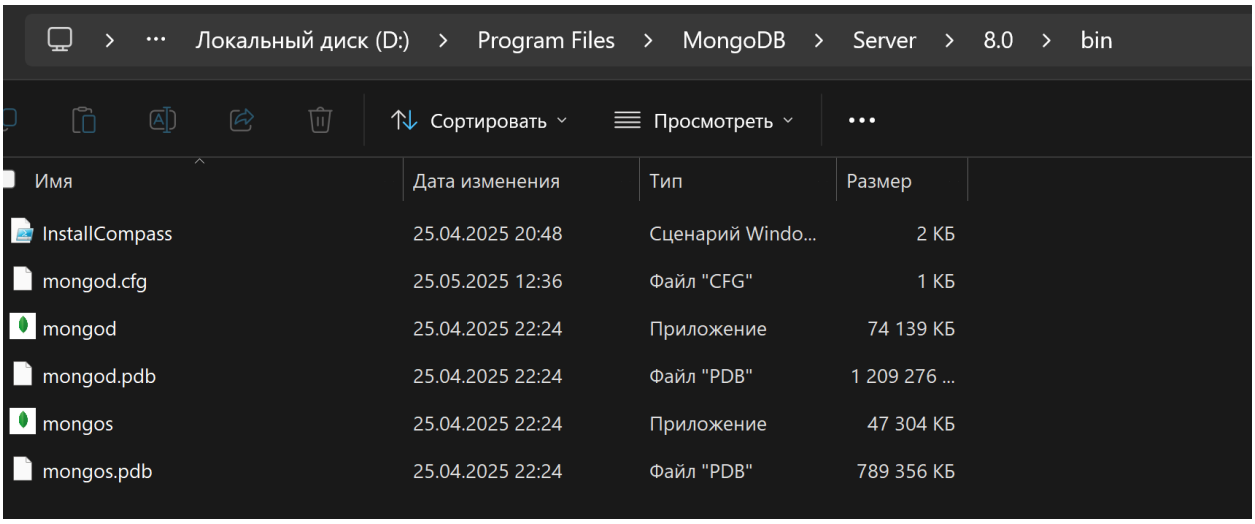


Рисунок 1. Файлы в папке bin

Аналог я запустил через MongoDB Compass, предварительно запустив mongod (сервер).

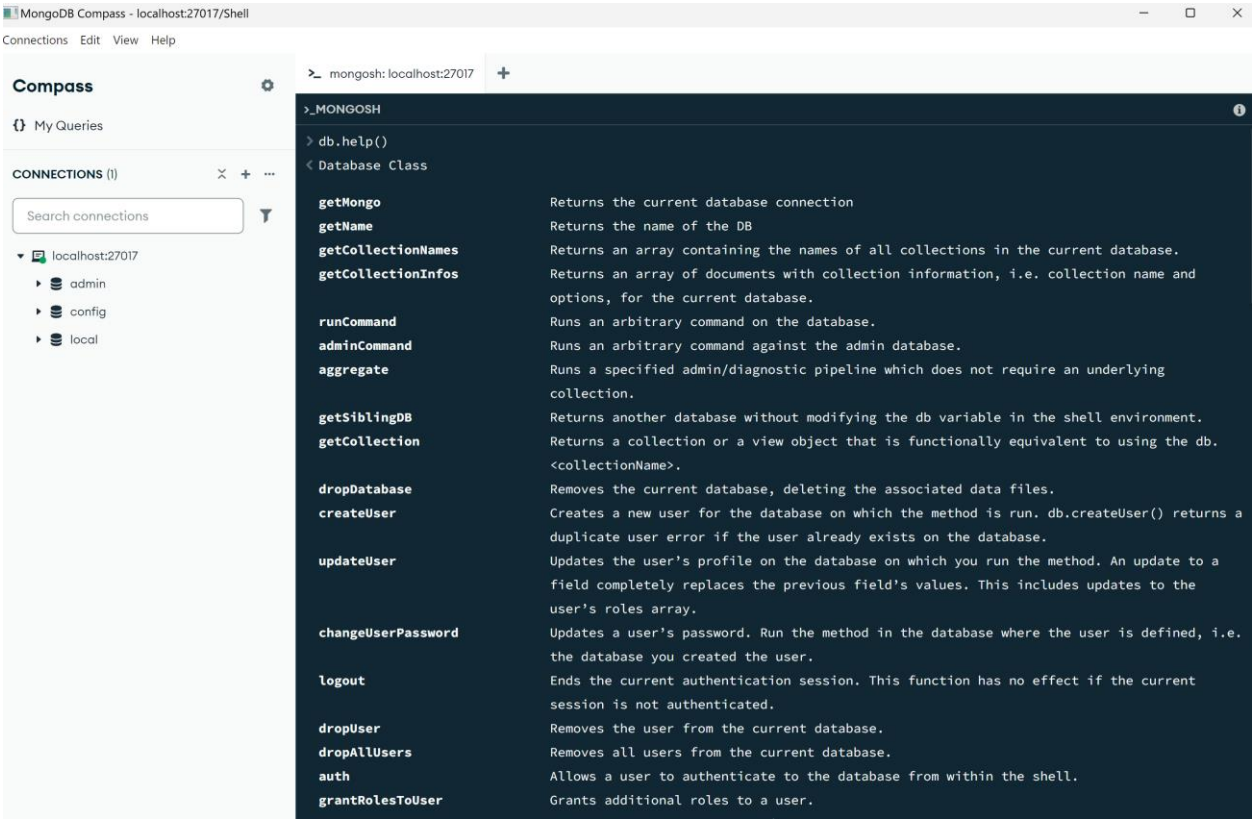


Рисунок 2. Вывод db.help() и общий вид консоли

Я запустил скрипты db.help(), db.help и db.stats(). Первые два вывели мне аналогичные данные.

```
> db.stats()
< {
    db: 'test',
    collections: Long('0'),
    views: Long('0'),
    objects: Long('0'),
    avgObjSize: 0,
    dataSize: 0,
    storageSize: 0,
    indexes: Long('0'),
    indexSize: 0,
    totalSize: 0,
    scaleFactor: Long('1'),
    fsUsedSize: 0,
    fsTotalSize: 0,
    ok: 1
  }
```

Рисунок 3. Вывод db.stats()

```
> db.help
< Database Class

  getMongo                  Returns the current database connection
  getName                   Returns the name of the DB
  getCollectionNames        Returns an array containing the names of all collections in the current database.
  getCollectionInfos        Returns an array of documents with collection information, i.e. collection name and
                            options, for the current database.
  runCommand                Runs an arbitrary command on the database.
  adminCommand              Runs an arbitrary command against the admin database.
  aggregate                 Runs a specified admin/diagnostic pipeline which does not require an underlying
                            collection.
  getSiblingDB              Returns another database without modifying the db variable in the shell environment.
  getCollection             Returns a collection or a view object that is functionally equivalent to using the db.
```

Рисунок 4. Вывод db.help

Далее идем по инструкции:

4. use learn

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

5. show dbs

```
> show dbs
< admin   40.00 KiB
  config  72.00 KiB
  local   40.00 KiB
learn>
```

6. вставка данных

```
> db.unicorns.insertOne({name: 'Aurora', gender: 'f', weight: 450})
< {
    acknowledged: true,
    insertedId: ObjectId('68333ecfd2b63cb4b6ed68d9')
  }
```

7. просмотр текущих коллекций

```
> show collections
< unicorns
learn>
```

8. переименование коллекции

```
> db.unicorns.renameCollection("MyUnicorns")
< { ok: 1 }
> show collections
< MyUnicorns
learn>
```

9. просмотр статистики

Тут выдало очень много информации, на скринах начало и содержательная часть

```
> db.MyUnicorns.stats()
< {
    ok: 1,
    capped: false,
    wiredTiger: {
      metadata: { formatVersion: 1 },
      creationString: 'access_pattern_hint=none,allocation_size=4KB,app_metadata=(formatVersion=1),assert=(commit_timesta
      type: 'file',
      uri: 'statistics:table:collection-7-9460511299300360945',
      LSM: {
        'bloom filter false positives': 0,
        'bloom filter hits': 0,
        'bloom filter misses': 0,
        'bloom filter pages evicted from cache': 0,
        'bloom filter pages read into cache': 0,
        'bloom filters in the LSM tree': 0,
        'chunks in the LSM tree': 0,
        'highest merge generation in the LSM tree': 0,
        'queries that could have benefited from a Bloom filter that did not exist': 0,
        'sleep for LSM checkpoint throttle': 0,
        'sleep for LSM merge throttle': 0,
        'total size of bloom filters': 0
      },
      autocommit: {
        'retries for readonly operations': 0,
        'retries for update operations': 0
```

10. удаление коллекции

11. удаление БД

Ход работы (часть 2)

2.1.1

1. Создаем базу данных аналогично 1 части работы.

2. Затем вставляем команды для заполнения данных:

```
> 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: 'Roooooodles', 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'});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('68334ff9d2b63cb4b6ed68e4')
    }
  }
> db.unicorns.find()
< {
    _id: ObjectId('68334ff9d2b63cb4b6ed68da'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
```

Так как у меня стоит 8 версия, выдалось предупреждение о том, что команда insert устарела, но при проверке видно, что все вставилось как надо.

3. Вставляем документ в коллекцию единорогов:

```
> document = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
< {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
> db.unicorns.insertOne(document)
< {
    acknowledged: true,
    insertedId: ObjectId('683352e3d2b63cb4b6ed68e5')
  }
learn>
```

4. Проверяем содержимое коллекции. Вывод длинный, поэтому на скринах начало и конец (с нашим новым единорогом)

```
> db.unicorns.find()
< {
    _id: ObjectId('68334ff9d2b63cb4b6ed68da'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('68334ff9d2b63cb4b6ed68db'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('68334ff9d2b63cb4b6ed68dc'),
    name: 'Unicrom',
    loves: [
      'energon',
```

```
  }
  {
    _id: ObjectId('68334ff9d2b63cb4b6ed68e4'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
  {
    _id: ObjectId('683352e3d2b63cb4b6ed68e5'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
learn>
```

2.2.1

1. Формируем необходимые запросы

Список самцов:

```
> db.unicorns.find({gender: 'm'})
< {
    _id: ObjectId('68334ff9d2b63cb4b6ed68da'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('68334ff9d2b63cb4b6ed68dc'),
```

Список самок:

```
> db.unicorns.find({gender: 'f'})
< {
    _id: ObjectId('68334ff9d2b63cb4b6ed68db'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

С ограничениями и фильтрами:

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
< {
    _id: ObjectId('68334ff9d2b63cb4b6ed68db'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('68334ff9d2b63cb4b6ed68df'),
    name: 'Ayna',
```

2. Самки, которые любят морковь (ограничение 2 способами)

```
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
< {
    _id: ObjectId('68334ff9d2b63cb4b6ed68db'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
< {
    _id: ObjectId('68334ff9d2b63cb4b6ed68db'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
learn>
```

2.2.2

Самцы отсортированы по имени, без предпочтений и поля _id.

```
> db.unicorns.find({gender: 'm'}, {loves: 0, '_id': 0}).sort({name: 1})
< {
    name: 'Dunx',
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    name: 'Horny',
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    name: 'Kenny',
    weight: 690,
    gender: 'm',
    vampires: 39
  }
  {
    name: 'Pilot',
    weight: 650,
    gender: 'm',
    vampires: 54
  }
  {
    name: 'Raleigh',
    weight: 421,
```

2.2.3

Список единорогов в обратном порядке добавления. Для этого используем $natural: -1

```
> db.unicorns.find().sort({$natural: -1})
< {
    _id: ObjectId('683352e3d2b63cb4b6ed68e5'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('68334ff9d2b63cb4b6ed68e4'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
  {
    _id: ObjectId('68334ff9d2b63cb4b6ed68e3'),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon'
    ],
```

2.2.4

```
> db.unicorns.find({}, {_id: 0, loves: {$slice : 1}})
< {
    name: 'Horny',
    loves: [
      'carrot'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    name: 'Aurora',
    loves: [
      'carrot'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    name: 'Unicrom',
    loves: [
      'energon'
    ],
    weight: 984,
```

2.3.1

```
> 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
  }
  {
```

2.3.2

```
> db.unicorns.find({gender: "m", weight: {$gte: 500}, loves: {$all: ["grape", "lemon"]}}, {_id: 0});
< {
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
learn>
```

2.3.3

```
> db.unicorns.find({vampires: {$exists: false}});
< {
    _id: ObjectId('68334ff9d2b63cb4b6ed68e4'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
learn>
```

2.3.4

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

С этого места я все-таки решил отдельно установить mongosh. Разницы особо нету, просто более короткие выводы иногда.

3.1.1

Таким образом заполняем новую коллекцию:

```
learn> db.towns.find()
[
  {
    _id: ObjectId('6835e47cde65b3191c6c4bd0'),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('6835e4c7de65b3191c6c4bd1'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('6835e4dfde65b3191c6c4bd2'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>
```

Ищем города с независимыми мэрами:

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

Ищем города с беспратийными мэрами (party отсутствует)

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

3.1.2

1. Функция для вывода списка самцов единорогов

```
function getMaleUnicorns() {
        return db.unicorns.find({gender:"m"});
}
```

2. Создать курсор для первых двух особей с сортировкой

```
 var cursor = db.unicorns.find({gender:"m"}).sort({name: 1}).limit(2);
```

3. Вывести результат используя forEach

```
cursor.forEach(function(doc){
        print(doc.name);
});
```

```
learn> function getMaleUnicorns() {
... return db.unicorns.find({gender:"m"});
... }
...
[Function: getMaleUnicorns]
learn> var cursor = db.unicorns.find({gender:"m"}).sort({name: 1}).limit(2);

learn> cursor.forEach(function(doc){
... print(doc.name);
... });
...
Dunx
Horny

learn>
```

3.2.1

```
learn> db.unicorns.count({gender: "f", weight: {$gte: 500, $lte: 600}});
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
2
learn> db.unicorns.countDocuments({gender: "f", weight: {$gte: 500, $lte: 600}});

2
learn>
```

Тут MongoDB ругался на count(), поэтому вместо него во второй раз попробовал countDocuments()

3.2.2

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

3.2.3

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

3.3.1

Тут save выдавало TypeError, поэтому было заменено с помощью replaceOne

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

3.3.2

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

3.3.3

```
learn> db.unicorns.updateOne({name: "Raleigh"}, {$push: {loves: "redbull"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>
```

```
learn> db.unicorns.find({name: "Raleigh"});
[
  {
    _id: ObjectId('68334ff9d2b63cb4b6ed68e1'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 7
  }
]
learn>
```

3.3.4

```
learn> db.unicorns.updateMany({gender: "m"}, {$inc: {vampires: 5}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
```

3.3.5

```
learn> db.towns.updateOne({name: "Portland"}, {$unset: {"mayor.party": ""}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({name: "Portland"});
[
  {
    _id: ObjectId('6835e4dfde65b3191c6c4bd2'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
learn>
```

3.3.6

```
learn> db.unicorns.updateOne({name: "Pilot"}, {$push: {loves: "chocolate"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.findOne({name: "Pilot"});
{
  _id: ObjectId('68334ff9d2b63cb4b6ed68e3'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon', 'chocolate' ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
learn>
```

3.3.7

```
learn> db.unicorns.updateOne({name: "Aurora"}, {$addToSet: {loves: {$each: ["sugar", "lemon"]}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.findOne({name: "Aurora"});
{
  _id: ObjectId('68334ff9d2b63cb4b6ed68db'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```

3.4.1

Удаляем беспартийных и проверяем:

```
learn> db.towns.deleteMany({"mayor.party": {$exists: false}});
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId('6835e4c7de65b3191c6c4bd1'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('6835e4dfde65b3191c6c4bd2'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>
```

Очищаем коллекцию и смотрим их список:

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

learn>
```

4.1.1

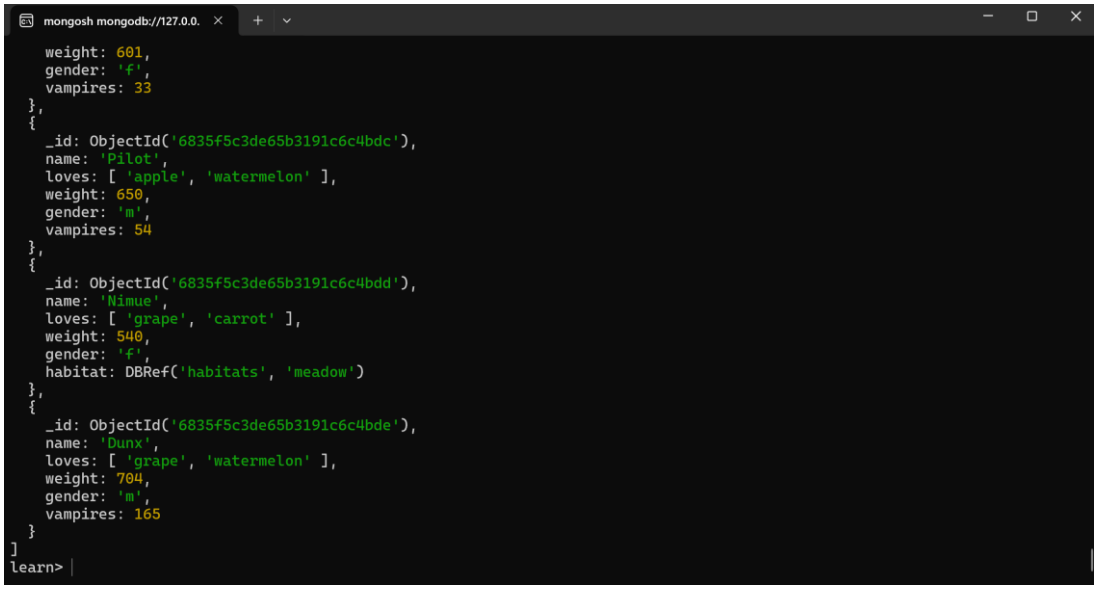
7. Создаем зоны обитания с кастомными _id:

```
db.habitats.insertMany([
{_id: "forest", name: "Dark Forest"},
{_id: "mountain", name: "High Mountains"},
{_id: "meadow", name: "Rainbow Meadows"} ]);
```

```
learn> db.habitats.insertMany([
... {_id: "forest", name: "Dark Forest"},
... {_id: "mountain", name: "High Mountains"},
... {_id: "meadow", name: "Rainbow Meadows"} ]);
...
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'mountain', '2': 'meadow' }
}
learn>
```

8. Включение ссылок на зоны обитания для единорогов (ручные ссылки) и проверка содержимого

```
learn> db.unicorns.updateOne({name: "Aurora"}, {$set: {habitat: {$ref: "habitats", $id: "forest"}}});
... db.unicorns.updateOne({name: "Unicrom"}, {$set: {habitat: {$ref: "habitats", $id: "mountain"}}});
... db.unicorns.updateOne({name: "Nimue"}, {$set: {habitat: {$ref: "habitats", $id: "meadow"}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('6835f5c3de65b3191c6c4bd3'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6835f5c3de65b3191c6c4bd4'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef('habitats', 'forest')
```



4.2.1

```
learn> db.unicorns.createIndex({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>
```

Вывод: можно, получилось

4.3.1

1. Получаем все индексы

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

2. Удаляем все индексы, кроме индекса для _id

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

3. Пробуем удалить индекс для _id

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

Получаем ошибку от сервера.

4.4.1

1. Создаем объемную коллекцию numbers

```
learn> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6835f9cbde65b3191c6dd27e') }
}
learn>
```

(Создавалась она довольно долго)

2. Выбираем последние 4 документа

```
learn> db.numbers.find().sort({value: -1}).limit(4);
[
  { _id: ObjectId('6835f9cbde65b3191c6dd27e'), value: 99999 },
  { _id: ObjectId('6835f9cbde65b3191c6dd27d'), value: 99998 },
  { _id: ObjectId('6835f9cbde65b3191c6dd27c'), value: 99997 },
  { _id: ObjectId('6835f9cbde65b3191c6dd27b'), value: 99996 }
]
learn>
```

3. Анализируем запрос 2

```
  queryHash: 'BA27D965',
  planCacheShapeHash: 'BA27D965',
  planCacheKey: '7A892B81',
  optimizationTimeMillis: 0,
  maxIndexedOrSolutionsReached: false,
  maxIndexedAndSolutionsReached: false,
  maxScansToExplodeReached: false,
  prunedSimilarIndexes: false,
  winningPlan: {
    isCached: false,
    stage: 'SORT',
    sortPattern: { value: -1 },
    memLimit: 104857600,
    limitAmount: 4,
    type: 'simple',
    inputStage: { stage: 'COLLSCAN', direction: 'forward' }
  },
  rejectedPlans: []
},
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 87,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
  executionStages: {
    isCached: false,
    stage: 'SORT',
    nReturned: 4,
    executionTimeMillisEstimate: 72,
    works: 100006,
    advanced: 4,
    needTime: 100001,
```

Получаем 87 миллисекунд

4. Создаем индекс для ключа value

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

5. Получаем информацию о всех индексах

```
learn> db.numbers.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
learn>
```

6. Выполняем еще раз запрос 2

```
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
learn> db.numbers.find().sort({value: -1}).limit(4);
[
  { _id: ObjectId('6835f9cbde65b3191c6dd27e'), value: 99999 },
  { _id: ObjectId('6835f9cbde65b3191c6dd27d'), value: 99998 },
  { _id: ObjectId('6835f9cbde65b3191c6dd27c'), value: 99997 },
  { _id: ObjectId('6835f9cbde65b3191c6dd27b'), value: 99996 }
]
learn>
```

7. Анализируем план выполнения запроса с установленным индексом

```
          isMultiKey: false,
          multiKeyPaths: { value: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'backward',
          indexBounds: { value: [ '[MaxKey, MinKey]' ] }
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 7,
    totalKeysExamined: 4,
    totalDocsExamined: 4,
    executionStages: {
      isCached: false,
      stage: 'LIMIT',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
```

8. Запрос с индексом занял 7 миллисекунд против 87 без него, что безусловно говорит об эффективности запроса с индексом.

Вывод

В ходе выполнения работы я познакомился с MongoDB и многими ее интересными свойствами. В частности, формат документов кажется мне удобным для некоторых случаев организации данных в не очень больших проектах. В то же время, последняя часть с индексами показывает, что для больших данных MongoDB тоже может подойти.