Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«ВВЕДЕНИЕ В СУБД MONGODB. УСТАНОВКА MONGODB. НАЧАЛО РАБОТЫ С БД»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся: Аплеев Дмитрий Артурович
Факультет прикладной информатики
Группа K3239
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2025

Цель работы:

Овладеть практическими навыками работы с MongoDB.

Ход работы

1. Установка

```
dimaapleev@MacBook-Air-Dima-2 ~ % mongod --version
db version v8.0.10
Build Info: {
    "version": "8.0.10",
    "gitVersion": "9d03076bb2d5147d5b6fe381c7118b0b0478b682",
    "modules": [],
    "allocator": "system",
    "environment": {
        "distarch": "aarch64",
        "target_arch": "aarch64"
    }
}
dimaapleev@MacBook-Air-Dima-2 ~ %
```

Выполним методы db.help(),db.stats()
db.help():

```
test> 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 d
atabase.
    aggregate                                 Runs a specified admin/diagnostic pipeline wh
ich does not require an underlying collection.
    getSiblingDB                              Returns another database without modifying th
e db variable in the shell environment.
    getCollection                             Returns a collection or a view object that is
 functionally equivalent to using the db.<collectionName>.
    dropDatabase                              Removes the current database, deleting the as
sociated data files.
    createUser                                Creates a new user for the database on which
the method is run. db.createUser() returns a duplicate user error if the user already exists
 on the database.
    updateUser                                Updates the user's profile on the database on
 which you run the method. An update to a field completely replaces the previous field's val
ues. This includes updates to the user's roles array.
    changeUserPassword                        Updates a user's password. Run the method in
the database where the user is defined, i.e. the database you created the user.
    logout                                    Ends the current authentication session. This
 function has no effect if the current session is not authenticated.
    dropUser                                  Removes the user from the current database.
    dropAllUsers                              Removes all users from the current database.
```

db.stats():

```
test> 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
}
test>
```

Создадим базу данных learn:

```
test> use learn
switched to db learn
learn>
```

Выведем список доступных БД:

```
learn> show dbs
admin   40.00 KiB
config  12.00 KiB
local   40.00 KiB
learn>
```

база learn пока не использовалась, поэтому она не отображается

Создание коллекции unicorns и просмотр

```
learn> db.unicorns.renameCollection('edinorojki')
{ ok: 1 }
learn> show collections
edinorojki
```

Переименование коллекции unicorns

```
learn> db.unicorns.renameCollection('edinorojki')
{ ok: 1 }
learn> show collections
edinorojki
```

Просмотр статистики коллекции

```
learn> db.edinorojki.stats()
{
  ok: 1,
  capped: false,
  wiredTiger: {
    metadata: { formatVersion: 1 },
    creationString: 'access_pattern_hint=none,allocation_size=4KB,app_metadata=(formatVersion=1),assert=(commit
_timestamp=none,durable_timestamp=none,read_timestamp=none,write_timestamp=off),block_allocation=best,block_com
pressor=snappy,cache_resident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0,encryption=(keyid=,n
ame=),exclusive=false,extractor=,format=btree,huffman_key=,huffman_value=,ignore_in_memory_cache_size=false,imm
utable=false,import=(compare_timestamp=oldest_timestamp,enabled=false,file_metadata=,metadata_file=,panic_corru
pt=true,repair=false),internal_item_max=0,internal_key_max=0,internal_key_truncate=true,internal_page_max=4KB,k
ey_format=q,key_gap=10,leaf_item_max=0,leaf_key_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true)
,lsm=(auto_throttle=true,bloom=true,bloom_bit_count=16,bloom_config=,bloom_hash_count=8,bloom_oldest=false,chun
k_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_custom=(prefix=,start_generation=0,suffix=),merge_max=15,me
rge_min=0),memory_page_image_max=0,memory_page_max=10m,os_cache_dirty_max=0,os_cache_max=0,prefix_compression=f
alse,prefix_compression_min=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=90,tiered_sto
rage=(auth_token=,bucket=,bucket_prefix=,cache_directory=,local_retention=300,name=,object_target_size=0),type=
file,value_format=u,verbose=[],write_timestamp_usage=none',
    type: 'file',
    uri: 'statistics:table:collection-7-8209857970208074697',
    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
    },
```

Удаление коллекции

```
learn> show collections
edinorojki
learn> db.edinorojki.drop()
true
learn> show collections

learn>
```

Удаление базы данных learn

```
learn> db.dropDatabase()
{ ok: 1, dropped: 'learn' }
learn> show dbs
admin   40.00 KiB
config  92.00 KiB
local   40.00 KiB
learn>
```

ЛАБОРАТОРНАЯ РАБОТА 6.2. Работа с БД в СУБД MongoDB
Задание 2.1.1

1. Создайте базу данных learn.
2. Заполните коллекцию единорогов unicorns:
3. Используя второй способ, вставьте в коллекцию единорогов документ.
4. Проверьте содержимое коллекции с помощью метода find.

Первый способ вставки:

```
test> use learn
switched to db learn
learn> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fb7503d8c22f97348698') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fb8003d8c22f97348699') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fb8803d8c22f9734869a') }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fb9703d8c22f9734869b') }
}
learn> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80})
;
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fba103d8c22f9734869c') }
}
learn> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fba803d8c22f9734869d') }
}
learn> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fbd403d8c22f9734869e') }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fbda03d8c22f9734869f') }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fbe603d8c22f973486a0') }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fbf703d8c22f973486a1') }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fc0003d8c22f973486a2') }
}
```

Способ вставки через переменную:

```
test> use learn
switched to db learn
learn> document=({name: "Dunx", loves: ['grape','watermelon'], weight:704, gender: 'm', vampires: 165})
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document)
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6843fd86390caaaef946a710') }
}
learn>
```

Проверка содержимого коллекции:

```
learn> db.unicorns.find({gender:'m'}).sort({name:1})
[
  {
    _id: ObjectId('6843fd86390caaaef946a710'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6843fb7503d8c22f97348698'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6843fbd403d8c22f9734869e'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6843fbf703d8c22f973486a1'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6843fbda03d8c22f9734869f'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('6843fb9703d8c22f9734869b'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6843fb8803d8c22f9734869a'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

2.2.1. Запросы для вывода списков самцов и самок

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

Список самцов, отсортированный по имени:

```
learn> db.unicorns.find({gender:'m'}).sort({name:1})
[
  {
    _id: ObjectId('6843fd86390caaaef946a710'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6843fb7503d8c22f97348698'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6843fbd403d8c22f9734869e'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6843fbf703d8c22f973486a1'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6843fbda03d8c22f9734869f'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('6843fb9703d8c22f9734869b'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6843fb8803d8c22f9734869a'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

Список самок, ограниченный первыми тремя, отсортированный по имени:

```
learn> db.unicorns.find({gender:'f'}).sort({name:1}).limit(3)
[
  {
    _id: ObjectId('6843fb8003d8c22f97348699'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6843fba803d8c22f9734869d'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('6843fbe603d8c22f973486a0'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

Самки, любящие carrot, ограниченные первой особью:

```
learn> db.unicorns.find({gender:'f', loves:'carrot'}).limit(1)
[
  {
    _id: ObjectId('6843fb8003d8c22f97348699'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn> db.unicorns.findOne({gender:'f', loves:'carrot'})
{
  _id: ObjectId('6843fb8003d8c22f97348699'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

2.2.2. Модификация запроса для самцов без loves и gender

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

Запрос для самцов без loves и gender:

```
learn> db.unicorns.find({gender:'m'}, {loves: 0, gender:0}).sort({name:1})
[
  {
    _id: ObjectId('6843fd86390caaaef946a710'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('6843fb7503d8c22f97348698'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('6843fbd403d8c22f9734869e'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('6843fbf703d8c22f973486a1'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('6843fbda03d8c22f9734869f'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('6843fb9703d8c22f9734869b'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('6843fb8803d8c22f9734869a'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
]
learn>
```

2.2.3. Вывод списка единорогов в обратном порядке добавления

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('6843fd86390caaaef946a710'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6843fc0003d8c22f973486a2'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6843fbf703d8c22f973486a1'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6843fbe603d8c22f973486a0'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('6843fbda03d8c22f9734869f'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
```

2.2.4. Вывод списка единорогов с первым loves, без _id

```
learn> 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,
    gender: 'm',
    vampires: 182
  },
  {
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
```

2.3.1 Вывод списка самок единорогов весом от полутонны до 700 кг, без _id

```
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'
  }
]
learn>
```

2.3.2. Вывод списка самцов единорогов весом от полутонны и предпочитающих grape и lemon, без _id

```
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
  }
]
learn>
```

2.3.3 Найти всех единорогов, не имеющих vampires

```
learn> db.unicorns.find({vampires: {$exists:false}})
[
  {
    _id: ObjectId('6843fc0003d8c22f973486a2'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn>
```

2.3.4 Вывод упорядоченного списка имён самцов единорогов с первым предпочтением

```
learn> 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' ] },
  { name: 'Pilot', loves: [ 'apple' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Roooooodles', loves: [ 'apple' ] },
  { name: 'Unicrom', loves: [ 'energon' ] }
]
learn>
```

3.1.1 Создание коллекции towns и выполнение запросов к вложенным объектам

Создание коллекции towns и вставка документов:

```
learn> db.towns.insertMany([{name: "Punxsutawney ",  populatiuon: 6200,  last_sensus: ISODate("2008-01-31"),  famou
s_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('68440ab2b2e7183626bb5641'),
    '1': ObjectId('68440ab2b2e7183626bb5642'),
    '2': ObjectId('68440ab2b2e7183626bb5643')
  }
}
learn> db.towns.find()
```

Запрос для городов с независимыми мэрами (party="I"):

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

Запрос для городов с беспартийными мэрами:

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

3.1.2 Создание функции и работа с курсором для единорогов

Создание функции для вывода списка самцов единорогов:

```
switched to db unicorns
unicorns> function MUnicorns() {return db.unicorns.find ({ gender: 'm'});}
[Function: MUnicorns]
```

Создание курсора для первых двух самцов с сортировкой по имени:

```
unicorns> var cursor = MUnicorns().sort({name:1}).limit(2)
```

Вывод результата:

```
learn> cursor.forEach(function (doc) {printjson(doc); });
{
  _id: ObjectId('6843fd86390caaaef946a710'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('6843fb7503d8c22f97348698'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}

learn>
```

3.2.1: вывести количество самок единорогов весом от полутонны до 600 кг.

```
learn> db.unicorns.find({gender: 'f', weight: {$gte:500, $lte:600}}).count()
(node:97641) [MONGODB DRIVER] Warning: cursor.count is deprecated and will be removed in the n
ext major version, please use `collection.estimatedDocumentCount` or `collection.countDocument
s` instead
(Use `node --trace-warnings ...` to show where the warning was created)
2
```

3.2.2: вывести список предпочтений.

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

3.2.3: посчитать количество особей единорогов.

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

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

```
learn> db.unicorns.updateOne({name:'Ayna'}, {$set:{weight:800, vampires:51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find(name:'Ayna')
Uncaught:
SyntaxError: Unexpected token, expected "," (1:21)

> 1 | db.unicorns.find(name:'Ayna')
    |                      ^
  2 |

learn> db.unicorns.find({name:'Ayna'})
[
  {
    _id: ObjectId('6843fba803d8c22f9734869d'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
learn>
```

3.3.4: Всем самцам единорогов увеличить количество убитых вапмиров на 5.

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

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

```
learn> db.towns.updateOne({name:'Portland'}, {$unset:{'mayor.party':1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({name:'Portland'})
[
  {
    _id: ObjectId('68440ab2b2e7183626bb5643'),
    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 Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.

```
learn> db.unicorns.updateOne({name:'Pilot'}, {$push:{loves:"chocolate"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name:'Pilot'})
[
  {
    _id: ObjectId('6843fbf703d8c22f973486a1'),
    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.

```
learn> db.unicorns.updateOne({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('6843fb8003d8c22f97348699'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

Практическое задание 3.4.1:

1. Создайте коллекцию towns
2. Удалите документы с беспартийными мэрами.
3. Проверьте содержание коллекции.
4. Очистите коллекцию.

Удаление беспартийных меров:

```
learn> db.towns.find()
[
  {
    _id: ObjectId('68440ab2b2e7183626bb5641'),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('68440ab2b2e7183626bb5642'),
    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('68440ab2b2e7183626bb5643'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
learn> db.towns.deleteMany({'mayor.party':{$exists:0}})
{ acknowledged: true, deletedCount: 2 }
learn> db.towns.find()
[
  {
    _id: ObjectId('68440ab2b2e7183626bb5642'),
    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' }
  }
]
```

Отчистка коллекции

```
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()

learn>
```

Практическое задание 4.1.1

Создание коллекции зон обитания

```
learn> db.zones.insertMany([{_id:'Лес', name:'Запретный лес',description:'Очень страшно'}, {_id:'Горы'
, name:'Уральские горы',description:'Очень высоко'}])
{ acknowledged: true, insertedIds: { '0': 'Лес', '1': 'Горы' } }
learn> show collections
towns
unicorns
zones
learn>
```

Добавление ссылки на зону

```
learn> db.unicorns.updateOne({name:'Horny'},{$set: {zone: {$ref:'zones', $id:'Лес'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne({name:'Aurora'},{$set: {zone: {$ref:'zones', $id:'Горы'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('6843fb7503d8c22f97348698'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    zone: DBRef('zones', 'Лес')
  },
  {
    _id: ObjectId('6843fb8003d8c22f97348699'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    zone: DBRef('zones', 'Горы')
  },
```

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

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

4.3.1

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

```
learn> db.unicorns.ensureIndex({name:1}, {unique:1})
[ 'name_1' ]
learn> db.unicorns.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
learn> db.unicorns.dropIndexe('_id_')
TypeError: db.unicorns.dropIndexe is not a function
learn>
```

4.4.1:

1. Создайте объемную коллекцию numbers, задействовав курсор:
2. for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
3. Выберите последних четыре документа.
4. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра executionTimeMillis)
5. Создайте индекс для ключа value.
6. Получите информацию о всех индексах коллекции numbres.
7. Выполните запрос 2.
8. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?
9. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

```
numbers> for (i=0; i<100000; i++){db.numbers.insert({value:i})}
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
```

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68441aa95e3238ce155ea495') }
}
```

```
numbers> db.numbers.find().sort({_id: -1}).limit(4)
[
  { _id: ObjectId('68441aa95e3238ce155ea495'), value: 99999 },
  { _id: ObjectId('68441aa95e3238ce155ea494'), value: 99998 },
  { _id: ObjectId('68441aa95e3238ce155ea493'), value: 99997 },
  { _id: ObjectId('68441aa95e3238ce155ea492'), value: 99996 }
]
```

```
numbers> db.numbers.explain('executionStats').find({value:99999})
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'numbers.numbers',
    parsedQuery: { value: { '$eq': 99999 } },
    indexFilterSet: false,
    queryHash: 'FBBD8DD0',
    planCacheShapeHash: 'FBBD8DD0',
    planCacheKey: 'CCC7FD70',
    optimizationTimeMillis: 2,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { value: { '$eq': 99999 } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 54,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { value: { '$eq': 99999 } },
      nReturned: 1,
      executionTimeMillisEstimate: 0,
      works: 100001,
      advanced: 1,
      needTime: 99999,
      needYield: 0,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 100000
    }
  },
  queryShapeHash: 'BB4A26650AA4F1DC930A665F7AE94318D9D8C81A0D15B0CA7695B37512548D58',
  command: { find: 'numbers', filter: { value: 99999 }, '$db': 'numbers' },
  serverInfo: {
    host: 'MacBook-Air-Dima-2.local',
    port: 27017,
    version: '8.0.10',
    gitVersion: '9d03076bb2d5147d5b6fe381c7118b0b0478b682'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
  },
  ok: 1
}
```

```
numbers> db.numbers.createIndex({value:1})
value_1
numbers> db.numbers.getIndex()
TypeError: db.numbers.getIndex is not a function
numbers> db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
numbers>
```

```
numbers> db.numbers.explain('executionStats').find({value:99999})
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'numbers.numbers',
    parsedQuery: { value: { '$eq': 99999 } },
    indexFilterSet: false,
    queryHash: 'FBBD8DD0',
    planCacheShapeHash: 'FBBD8DD0',
    planCacheKey: '463AB5A3',
    optimizationTimeMillis: 3,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { value: 1 },
        indexName: 'value_1',
        isMultiKey: false,
        multiKeyPaths: { value: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { value: [ '[99999, 99999]' ] }
      }
    },
    rejectedPlans: []
```

Запрос выполнился моментально

Вывод

В процессе выполнения лабораторной работы были изучены основные операции в MongoDB. Получены навыки работы с коллекциями.