ЛАБОРАТОРНАЯ РАБОТА 6

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

Цель: овладеть практическими навыками установки СУБД MongoDB.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 8.0.9 (последняя)

Практическое задание 1:

Я скачал monogodb с официального сайта, распоковал файлы и теперь могу проверить работоспособность системы с помощью базового метода 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 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.<collectionName>.
    dropDatabase                            Removes the current database, deleting the associated data files.
    createUser                              Creates a new user for the database on which the method is run. db.createUser() returns a duplicate user erro
r 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 repla
ces the previous field's values. 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 cr
eated the user.
    logout                                  Ends the current authentication session. This function has no effect if the current session is not authentica
ted.
    dropUser                                Removes the user from the current database.
    dropAllUsers                            Removes all users from the current database.
    auth                                    Allows a user to authenticate to the database from within the shell.
    grantRolesToUser                        Grants additional roles to a user.
    revokeRolesFromUser                     Removes a one or more roles from a user on the current database.
    getUser                                 Returns user information for a specified user. Run this method on the user's database. The user must exist on
 the database on which the method runs.
    getUsers                                Returns information for all the users in the database.
    createCollection                        Create new collection
    createEncryptedCollection               Creates a new collection with a list of encrypted fields each with unique and auto-created data encryption ke
ys (DEKs). This is a utility function that internally utilises ClientEnryption.createEncryptedCollection.
```

db.help имеет тот же вывод, а db.stats() выводит нулевые значения

```
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
}
```

Создадим БД learn

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

Добавим в неё коллекцию unicorns

```
learn> db.unicorns.insertOne({name: 'Aurora', gender: 'f', weight: 450})
{
  acknowledged: true,
  insertedId: ObjectId('682ebcc83768fb5d646c4bd0')
}
```

Посмотрим список коллекция

```
learn> show collections
unicorns
```

Переименуем коллекцию

```
learn> db.unicorns.renameCollection("people")
{ ok: 1 }
```

Посмотрим на статистику БД

```
learn> db.people.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_t
imestamp=none,write_timestamp=off),block_allocation=best,block_compressor=snappy,cache_resident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0
,encryption=(keyid=,name=),exclusive=false,extractor=,format=btree,huffman_key=,huffman_value=,ignore_in_memory_cache_size=false,immutable=false,import=(com
pare_timestamp=oldest_timestamp,enabled=false,file_metadata=,metadata_file=,panic_corrupt=true,repair=false),internal_item_max=0,internal_key_max=0,internal
_key_truncate=true,internal_page_max=4KB,key_format=q,key_gap=10,leaf_item_max=0,leaf_key_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),ls
m=(auto_throttle=true,bloom=true,bloom_bit_count=16,bloom_config=,bloom_hash_count=8,bloom_oldest=false,chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,me
rge_custom=(prefix=,start_generation=0,suffix=),merge_max=15,merge_min=0),memory_page_image_max=0,memory_page_max=10m,os_cache_dirty_max=0,os_cache_max=0,pr
efix_compression=false,prefix_compression_min=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=90,tiered_storage=(auth_token=,bucket=,b
ucket_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-7-1797651161612286 5286',
  sharded: false,
  size: 65,
  count: 1,
  numOrphanDocs: 0,
  storageSize: 20480,
  totalIndexSize: 20480,
  totalSize: 40960,
  indexSizes: { _id_: 20480 },
  avgObjSize: 65,
  ns: 'learn.people',
  nindexes: 1,
  scaleFactor: 1
```

(полностью характеристики модуля wiredTiger не были включены из-за громоздкости и незначимости [все значения параметров нули]). Удалим коллекцию

```
learn> db.people.drop()
true
```

Удалим базу данных

```
learn> db.dropDatabase()
{ ok: 1, dropped: 'learn' }
```

Практическое задание 2.1.1

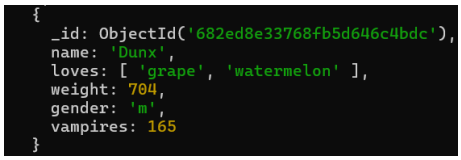
Добавим в базу данных коллекции единорогов unicorns

```
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: '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('682ed5d83768fb5d646c4bdb') }
}
```

Добавим колекцию как документ

```
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)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('682ed8e33768fb5d646c4bdc') }
}
```

Проверим содержимое коллекции и обнаружим там запись Dunx



Практическое задание 2.2

Выведем список самцов, отсортировав списки по имени

Выведем списки самок, ограничив список самок первыми 3 особями, и отсортировав списки по имени

Вывести список самок, которые любят carrot, ограничив список с помощью findOne

Вывести список самок, которые любят carrot, ограничив список с помощью limit

```
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd2'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

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

```
learn> db.unicorns.find({ gender: 'm' },{ loves: 0, vampires: 0 })
[
  {
    _id: ObjectId('682ed5d73768fb5d646c4bd1'),
    name: 'Horny',
    weight: 600,
    gender: 'm'
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd3'),
    name: 'Unicrom',
    weight: 984,
    gender: 'm'
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd4'),
    name: 'Roooooodles',
    weight: 575,
    gender: 'm'
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd7'),
    name: 'Kenny',
    weight: 690,
    gender: 'm'
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd8'),
    name: 'Raleigh',
    weight: 421,
    gender: 'm'
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bda'),
    name: 'Pilot',
    weight: 650,
    gender: 'm'
  },
```

Выведем список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({ _id: -1 })
[
  {
    _id: ObjectId('682ed8e33768fb5d646c4bdc'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bdb'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bda'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd9'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd8'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd7'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd6'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd5'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd4'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('682ed5d83768fb5d646c4bd3'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
```

Выведем список единорогов с названием первого любимого предпочтения, исключив идентификатор

```
learn> db.unicorns.find({}, { _id: 0, name: 1, gender: 1,weight: 1, firstLove: { $slice: ["$loves", 1] }})
[
  { name: 'Horny', weight: 600, gender: 'm', firstLove: [ 'carrot' ] },
  { name: 'Aurora', weight: 450, gender: 'f', firstLove: [ 'carrot' ] },
  {
    name: 'Unicrom',
    weight: 984,
    gender: 'm',
    firstLove: [ 'energon' ]
  },
  {
    name: 'Roooooodles',
    weight: 575,
    gender: 'm',
    firstLove: [ 'apple' ]
  },
  { name: 'Solnara', weight: 550, gender: 'f', firstLove: [ 'apple' ] },
  {
    name: 'Ayna',
    weight: 733,
    gender: 'f',
    firstLove: [ 'strawberry' ]
  },
  { name: 'Kenny', weight: 690, gender: 'm', firstLove: [ 'grape' ] },
  { name: 'Raleigh', weight: 421, gender: 'm', firstLove: [ 'apple' ] },
  { name: 'Leia', weight: 601, gender: 'f', firstLove: [ 'apple' ] },
  { name: 'Pilot', weight: 650, gender: 'm', firstLove: [ 'apple' ] },
  { name: 'Nimue', weight: 540, gender: 'f', firstLove: [ 'grape' ] },
  { name: 'Dunx', weight: 704, gender: 'm', firstLove: [ 'grape' ] }
]
```

Практическое задание 2.3:

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: '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'
  }
]
```

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

```
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
  }
]
```

Найти всех единорогов, не имеющих ключ vampires

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

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

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

Практическое задание 3.1

Создадим коллекцию towns

```
learn> 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: ["statue 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('682f5fe742eb907bd66c4be7'),
    '1': ObjectId('682f5fe742eb907bd66c4be8'),
    '2': ObjectId('682f5fe742eb907bd66c4be9')
  }
}
```

Вернём список городов с независимыми мэрами (party="I"). (только название города и информацию о мэре)

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

Вернём список беспартийных мэров (только название города и информацию о мэре)

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

Напишем функцию для вывода списка самцов единорогов.

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

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

```
learn> var maleUnicornsCursor = getMaleUnicorns()

learn> maleUnicornsCursor.forEach(function(unicorn) {printjson({name: unicorn.name, loves: unicorn.loves, weight: unicorn.weight, vampires: unicorn.vampires
 || 'не указано'});});
{
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  vampires: 165
}
{
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  vampires: 63
}
```


Практическое задание 3.2

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Вывести список предпочтений.

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

Посчитать количество особей единорогов обоих полов.

```
learn> db.unicorns.aggregate([{$group: {_id: "$gender", count: {$sum: 1}}}, {$project: {gender: "$_id", count: 1,_id: 0}}])
[ { count: 7, gender: 'm' }, { count: 5, gender: 'f' } ]
```

Практическое задание 3.3

Добавим нового единорога

```
learn> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedId: ObjectId('682f68c642eb907bd66c4bf6')
}
```

Посмотрим на коллекцию unicorns (Barny появился)

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

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

Посмотрим на обновлённую запись единорога Ayna

```
learn> db.unicorns.find({ name: "Ayna" }).pretty()
[
  {
    _id: ObjectId('682f636c42eb907bd66c4bef'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

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

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

Посмотрим на обновлённую запись

```
learn> db.unicorns.find({ name: "Raleigh" }).pretty()
[
  {
    _id: ObjectId('682f636c42eb907bd66c4bf1'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Всем самцам единорогов увеличим количество убитых вапмиров на 5.

```
learn> db.unicorns.updateMany({gender: "m"},{$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
```

Посмотрим на обновлённую запись

```
learn> db.unicorns.find({ gender: "m" }, { name: 1, vampires: 1, _id: 0 }).pretty()
[
  { name: 'Horny', vampires: 68 },
  { name: 'Unicrom', vampires: 187 },
  { name: 'Roooooodles', vampires: 104 },
  { name: 'Kenny', vampires: 44 },
  { name: 'Raleigh', vampires: 7 },
  { name: 'Pilot', vampires: 59 },
  { name: 'Dunx', vampires: 170 },
  { name: 'Barny', vampires: 5 }
]
```

Изменим информацию о городе Портланд: мэр этого города теперь беспартийный.

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

Посмотрим на обновлённую запись

```
learn> db.towns.find({name: "Portland"}).pretty()
[
  {
    _id: ObjectId('682f5fe742eb907bd66c4be9'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

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

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

Посмотрим на обновлённую запись

```
learn> db.unicorns.find({ name: "Pilot" }).pretty()
[
  {
    _id: ObjectId('682f636c42eb907bd66c4bf3'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

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

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

Посмотрим на обновлённую запись

```
learn> db.unicorns.find({ name: "Aurora" }).pretty()
[
  {
    _id: ObjectId('682f636c42eb907bd66c4beb'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 3.4

Создим коллекцию towns, включающую заданные документы

```
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: ["statue 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('682f723b42eb907bd66c4bf7'),
    '1': ObjectId('682f723b42eb907bd66c4bf8'),
    '2': ObjectId('682f723b42eb907bd66c4bf9')
  }
}
```

Удалим документы с беспартийными мэрами.

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

Проверим содержание коллекции.

```
learn> db.towns.find().pretty()
[
  {
    _id: ObjectId('682f723b42eb907bd66c4bf8'),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('682f723b42eb907bd66c4bf9'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

Очистим коллекцию.

```
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 2 }
```

Просмотрим список доступных коллекций.

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

(коллекция осталось, но она теперь пустая)

Практическое задание 4

Создадим коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.

```
learn> db.habitats.insertMany([
...   {
...     _id: "mf",
...     fullName: "Magic Forest",
...     description: "Густой волшебный лес с древними деревьями"
...   },
...   {
...     _id: "cv",
...     fullName: "Crystal Valley",
...     description: "Долина кристальных водопадов и чистых озер"
...   },
...   {
...     _id: "em",
...     fullName: "Enchanted Mountains",
...     description: "Зачарованные горы с вечными снегами"
...   }
... ])
{
  acknowledged: true,
  insertedIds: { '0': 'mf', '1': 'cv', '2': 'em' }
}
```

Включим для нескольких единорогов в документы ссылку на зону обитания, использую второй способ автоматического связывания

```
learn> db.unicorns.updateOne(
...   {name: "Horny"},
...   {$set: {habitat: {$ref: "habitats", $id: "mf"}}}
... )
... db.unicorns.updateOne(
...   {name: "Aurora"},
...   {$set: {habitat: {$ref: "habitats", $id: "cv"}}}
... )
... db.unicorns.updateOne(
...   {name: "Unicrom"},
...   {$set: {habitat: {$ref: "habitats", $id: "em"}}}
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Проверим содержание коллекции едиорогов.

```
learn> db.unicorns.find({},{name: 1, habitat: 1, _id: 0}).pretty()
[
  { name: 'Horny', habitat: DBRef('habitats', 'mf') },
  { name: 'Aurora', habitat: DBRef('habitats', 'cv') },
  { name: 'Unicrom', habitat: DBRef('habitats', 'em') },
```

Проверим, можно ли задать уникальный идентификатор по имени

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

Оказывается, что можно. Получим информацию о всех индексах коллекции unicorns

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

Удалим все индексы, кроме индекса для идентификатора.

```
learn> db.unicorns.dropIndex("name_1")
{ nIndexesWas: 2, ok: 1 }
```

Попытаемся удалить индекс для идентификатора.

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

Очевидно, мы получаем ошибку, так как таблица без идентификатора существовать не может. Создадим объемную коллекцию numbers, задействовав курсор:

```
learn> for (i = 0; i < 100000; i++) {db.numbers.insertOne({value: i})}
{
  acknowledged: true,
  insertedId: ObjectId('682f798c42eb907bd66e3cc5')
}
```

Выберем последних четыре документа (2)

```
learn> db.numbers.find().sort({value: -1}).limit(4)
[
  { _id: ObjectId('682f798c42eb907bd66e3cc5'), value: 99999 },
  { _id: ObjectId('682f798c42eb907bd66e3cc4'), value: 99998 },
  { _id: ObjectId('682f798c42eb907bd66e3cc3'), value: 99997 },
  { _id: ObjectId('682f798c42eb907bd66e3cc2'), value: 99996 }
]
```

Проанализируем план выполнения запроса 2 и узнаем сколько потребовалось времени на выполнение запроса?

```
learn> var explain = db.numbers.find().sort({value: -1}).limit(4).explain("executionStats")
... print("Время выполнения без индекса (мс): " + explain.executionStats.executionTimeMillis)
Время выполнения без индекса (мс): 87
```

Создадим индекс для ключа value.

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

Получим информацию о всех индексах коллекции numbres.

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

Выполните запрос 2

```
learn> db.numbers.find().sort({value: -1}).limit(4)
[
  { _id: ObjectId('682f798c42eb907bd66e3cc5'), value: 99999 },
  { _id: ObjectId('682f798c42eb907bd66e3cc4'), value: 99998 },
  { _id: ObjectId('682f798c42eb907bd66e3cc3'), value: 99997 },
  { _id: ObjectId('682f798c42eb907bd66e3cc2'), value: 99996 }
]
```

Проанализируем план выполнения запроса с установленным индексом и узнаем сколько потребовалось времени на выполнение запроса?

```
learn> var explainIndexed = db.numbers.find().sort({value: -1}).limit(4).explain("executionStats")
... print("Время выполнения с индексом (мс): " + explainIndexed.executionStats.executionTimeMillis)
Время выполнения с индексом (мс): 10
```

То есть без индексов запрос выполняется за 87 мс, а с индексами за 10 мс (почти в 9 раз быстрее). Значит, запрос с индексом более эффективен