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

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«РАБОТА С БД В СУБД MongoDB»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Смирновой Карины
Факультет прикладной информатики
Группа К3241
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

Цель работы: овладеть практическими навыками установки СУБД MongoDB.

ВЫПОЛНЕНИЕ 6.1

Практическое задание:

1. Установите MongoDB для обоих типов систем (32/64 бита).
2. Проверьте работоспособность системы запуском клиента mongo.
3. Выполните методы:
    1. 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 d
atabase.
    getCollectionInfos                      Returns an array of documents with collection information, i.e. collectio
n 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 unde
rlying collection.
    getSiblingDB                            Returns another database without modifying the db variable in the shell e
nvironment.
    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.create
User() returns a duplicate user error if the user already exists on the database.
```

    2. 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 d
atabase.
    getCollectionInfos                      Returns an array of documents with collection information, i.e. collectio
n 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 unde
```

    3. 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
}
```

4. Создайте БД learn.

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

5. Получите список доступных БД.

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

6. Создайте коллекцию unicorns, вставив в нее документ {name: 'Aurora', gender: 'f', weight: 450}.

```
learn> db.learn.insertOne({name: 'Aurora', gender: 'f', weight: 450})
{
  acknowledged: true,
  insertedId: ObjectId('681df1d104dee6df406c4bd0')
}
learn>
```

7. Просмотрите список текущих коллекций.

```
learn> show collections
learn
learn>
```

8. Переименуйте коллекцию unicorns.

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

9. Просмотрите статистику коллекции.

```
learn> db.unicorn.stats()
{
  ok: 1,
  capped: false,
  wiredTiger: {
    metadata: { formatVersion: 1 },
    creationString: 'access_pattern_hint=none,allocation_size=4KB,app_me
p=none,durable_timestamp=none,read_timestamp=none,write_timestamp=off),b
he_resident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0
ctor=,format=btree,huffman_key=,huffman_value=,ignore_in_memory_cache_si
mp=oldest_timestamp,enabled=false,file_metadata=,metadata_file=,panic_co
ernal_key_max=0,internal_key_truncate=true,internal_page_max=4KB,key_for
leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_thro
nfig=,bloom_hash_count=8,bloom_oldest=false,chunk_count_limit=0,chunk_ma
t_generation=0,suffix=),merge_max=15,merge_min=0),memory_page_image_max=
che_max=0,prefix_compression=false,prefix_compression_min=4,source=,spli
lit_pct=90,tiered_storage=(auth_token=,bucket=,bucket_prefix=,cache_dire
ize=0),type=file,value_format=u,verbose=[],write_timestamp_usage=none',
    type: 'file',
    uri: 'statistics:table:collection-7-14436298675731420906',
    LSM: {
      'bloom filter false positives': 0
```

10. Удалите коллекцию.

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

11. Удалите БД learn.

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

ВЫПОЛНЕНИЕ 6.2

Практическое задание 2.1.1:

1. Создайте базу данных learn.

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

2. Заполните коллекцию единорогов unicorns

3. Используя второй способ, вставьте в коллекцию единорогов 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('681df929bf0f62d1e96c4bdb') }
}
learn>
```

4. Проверьте содержимое коллекции с помощью метода find.

db.unicorns.find()

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('681df85fbf0f62d1e96c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('681df86abf0f62d1e96c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('681df875bf0f62d1e96c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
```

Практическое задание 2.2.1

1. Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.

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

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('681df929bf0f62d1e96c4bdb'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('681df85fbf0f62d1e96c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
```

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('681df86abf0f62d1e96c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('681df897bf0f62d1e96c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('681df8acbf0f62d1e96c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn>
```

2. Найдите всех самок, которые любят carrot. Ограничьте этот список первой особью с помощью функций findOne и limit.

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

```
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId('681df86abf0f62d1e96c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 2.2.2:

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

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).sort({name: 1
[
  {
    _id: ObjectId('681df929bf0f62d1e96c4bdb'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('681df85fbf0f62d1e96c4bd0'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('681df89ebf0f62d1e96c4bd6'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
```

Практическое задание 2.2.3:

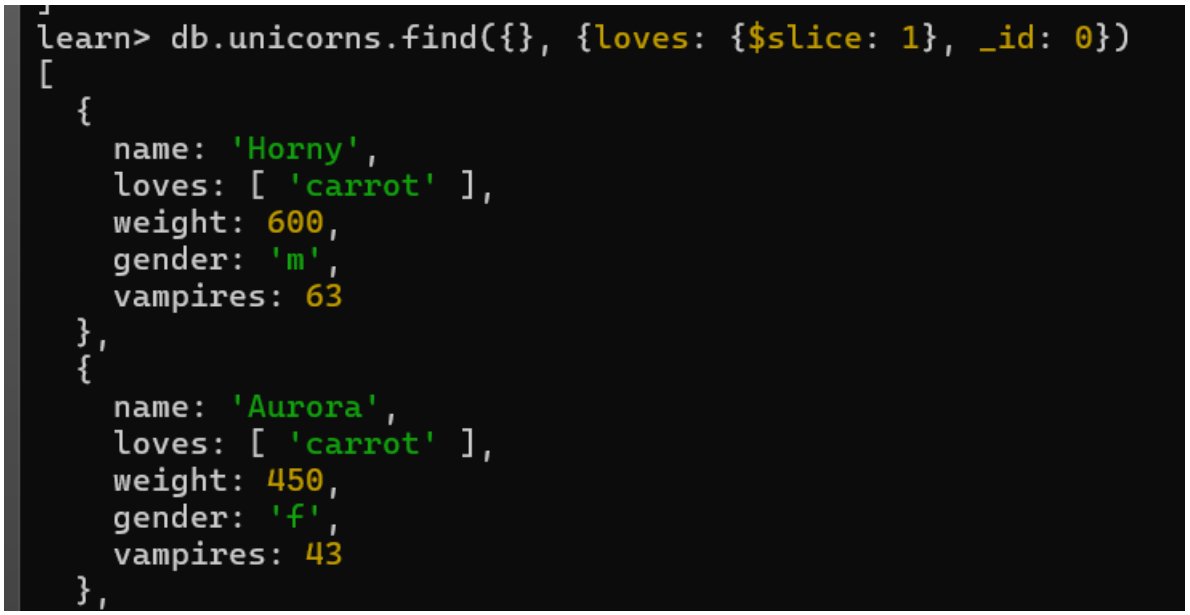
Вывести список единорогов в обратном порядке добавления.

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

Практическое задание 2.1.4:

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

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



Практическое задание 2.3.1:

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

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

```
learn> db.unicorns.find({ gender: 'f', weight: { $gt: 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'
  }
]
```

Практическое задание 2.3.2:

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

db.unicorns.find({ gender: 'm', weight: { $gt: 500}, loves : {$all: ['grape', 'lemon']} },{ _id: 0 })

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

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

db.unicorns.find({vampires: {$exists: false}})

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('681df8bcbf0f62d1e96c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.4:

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

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

```
learn> db.unicorns.find({gender: 'm'}, {name: 1, _id: 0, 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' ] }
]
```

Практическое задание 3.1.1:

Создайте коллекцию towns, включающую следующие документы

```
{name: "Punxsutawney ",
populatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: [""],
mayor: {
   name: "Jim Wehrle"
   }}

{name: "New York",
populatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
   name: "Michael Bloomberg",
   party: "I"}}

{name: "Portland",
populatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
   name: "Sam Adams",
   party: "D"}}
```

Сформировать запрос, который возвращает список городов с независимыми мэрами (party="I"). Вывести только название города и информацию о мэре.

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

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

Сформировать запрос, который возвращает список беспартийных мэров (party отсутствует). Вывести только название города и информацию о мэре.

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

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

Практическое задание 3.1.2:

Сформировать функцию для вывода списка самцов единорогов.

const findMaleUnicorn = () => db.unicorns.find({gender: 'm'});

```
learn> const findMaleUnicorn = () => db.unicorns.find({gender: 'm'});
```

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

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

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

Вывести результат, используя forEach.

cursor.forEach(function(obj) {

print(obj.name);

})

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

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

db.unicorns.find({gender: 'f', weight: {$gt: 500, $lt: 600}}).count()

```
learn> db.unicorns.find({gender: 'f', weight: {$gt: 500, $lt: 600}}).count()
2
learn>
```

Практическое задание 3.2.2:

Вывести список предпочтений.

db.unicorns.distinct("loves")

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

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

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

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

Практическое задание 3.3.1:

Выполнить команду:

> db.unicorns.save({name: 'Barny', loves: ['grape'],

weight: 340, gender: 'm'})

Проверить содержимое коллекции unicorns.

Метод save() больше не поддерживается.

Практическое задание 3.3.2:

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

Проверить содержимое коллекции unicorns.

db.unicorns.updateOne({name: 'Ayna'}, {$set: {weight: 800, vampires: 51}}, {upsert: true})

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

Практическое задание 3.3.3:

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

Проверить содержимое коллекции unicorns.

db.unicorns.updateOne({name: 'Raleigh'}, {$set: {loves: ['redbul']}}, {upsert: true})

```
learn> db.unicorns.updateOne({name: 'Raleigh'}, {$set: {loves: ['redbul']}}, {upsert: true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Практическое задание 3.3.4:

Всем самцам единорогов увеличить количество убитых вапмиров на 5.

Проверить содержимое коллекции unicorns.

db.unicorns.update({gender: 'm'}, {$inc: {vampires: 5}})

```
learn> db.unicorns.update({gender: 'm'}, {$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Практическое задание 3.3.5:

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

Проверить содержимое коллекции towns.

db.towns.updateOne({name: 'Portland'}, {$unset: {'mayor.party': 1}})

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

Практическое задание 3.3.6:

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

Проверить содержимое коллекции unicorns.

db.unicorns.update({name: 'Pilot'}, {$push: {loves: 'chocolate'}})

```
learn> db.unicorns.update({name: 'Pilot'}, {$push: {loves: 'chocolate'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>
```

Практическое задание 3.3.7:

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

Проверить содержимое коллекции unicorns.

db.unicorns.update({name: 'Aurora'}, {$addToSet: {loves: {$each: ['lemon, shugar']}}})

```
learn> db.unicorns.update({name: 'Aurora'}, {$addToSet: {loves: {$each: ['lemon, shugar']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>
```

Практическое задание 3.4.1:

Создайте коллекцию towns, включающую следующие документы:

```
{name: "Punxsutawney ",
popujatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: ["phil the groundhog"],
mayor: {
   name: "Jim Wehrle"
   }}

{name: "New York",
popujatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
   name: "Michael Bloomberg",
   party: "I"}}

{name: "Portland",
popujatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
   name: "Sam Adams",
   party: "D"}}
```

Удалите документы с беспартийными мэрами.

db.towns.remove({'mayor.party': {$exists: false}})

```
learn> db.towns.remove({'mayor.party': {$exists: false}})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 3 }
```

Проверьте содержание коллекции.

```
learn> db.towns.find()
[
  {
    _id: ObjectId('681e0d30bf0f62d1e96c4bdd'),
    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('681e2f59bf0f62d1e96c4be1'),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('681e2f6fbf0f62d1e96c4be2'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>
```

Очистите коллекцию.

```
learn> db.towns.remove({})
{ acknowledged: true, deletedCount: 3 }
learn>
```

Просмотрите список доступных коллекций.

```
learn> show collections
towns
unicorns
learn>
```

Практическое задание 4.1.1:

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

```
db.habitats.insertMany([
...   {
...     _id: "FR",
...     name: "France – Lavender Fields",
...     description: "Calm, floral meadows with soft hills."
...   },
...   {
...     _id: "JP",
...     name: "Japan – Sakura Highlands",
```

```
...     description: "Cherry blossom valleys with fresh springs."
...   },
...   {
...     _id: "BR",
...     name: "Brazil – Misty Rainforest",
...     description: "Humid jungle with glowing flora."
...   }
... ])
```

```
learn> db.habitats.insertMany([
...   {
...     _id: "FR",
...     name: "France – Lavender Fields",
...     description: "Calm, floral meadows with soft hills."
...   },
...   {
...     _id: "JP",
...     name: "Japan – Sakura Highlands",
...     description: "Cherry blossom valleys with fresh springs."
...   },
...   {
...     _id: "BR",
...     name: "Brazil – Misty Rainforest",
...     description: "Humid jungle with glowing flora."
...   }
... ])
...
{
  acknowledged: true,
  insertedIds: { '0': 'FR', '1': 'JP', '2': 'BR' }
}
learn>
```

2. Включите для нескольких единорогов в документы ссылку на зону обитания, использую второй способ автоматического связывания.

db.unicorns.update({name: 'Ayna'}, {$set: {habitats: {$ref: 'habitats', $id: 'FR'}}})

```
learn> db.unicorns.update({name: 'Ayna'}, {$set: {habitats: {$ref: 'habitats', $id: 'FR'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
learn> db.unicorns.update({name: 'Horny'}, {$set: {habitats: {$ref: 'habitats', $id: 'JP'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
learn> db.unicorns.update({name: 'Aurora'}, {$set: {habitats: {$ref: 'habitats', $id: 'BR'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

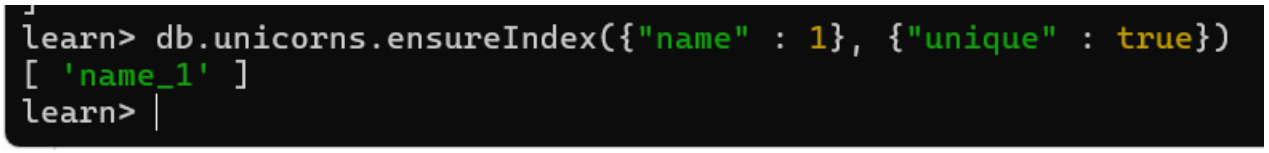
3. Проверьте содержание коллекции единорогов.

db.unicorns.find({ name: { $in: ["Ayna", "Horny", "Aurora"] } })

Практическое задание 4.2.1:

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

db.unicorns.ensureIndex({"name" : 1}, {"unique" : true})



Практическое задание 4.3.1:

1. Получите информацию о всех индексах коллекции unicorns .

db.unicorns.getIndexes()

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

2. Удалите все индексы, кроме индекса для идентификатора.

db.unicorns.dropIndex("name_1")

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

3. Попытайтесь удалить индекс для идентификатора.

db.unicorns.dropIndex("_id_")

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

Практическое задание 4.4.1:

Создайте объемную коллекцию numbers, задействовав курсор:

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

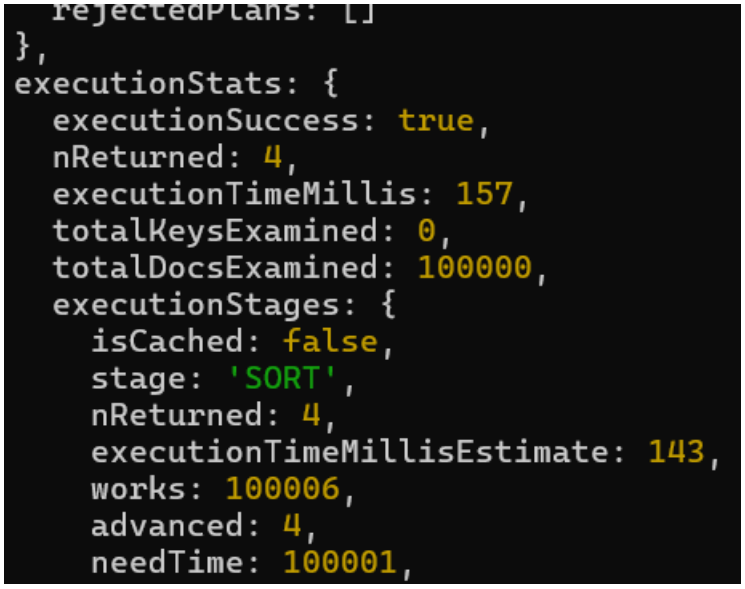
Выберите последних четыре документа.

Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра executionTimeMillis)

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

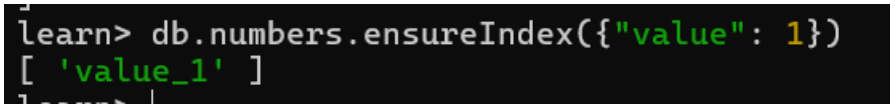
```
{ nIndexesWas: 2, ok: 1 }
learn> db.numbers.find().sort({ value: -1 }).limit(4).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'BA27D965',
    planCacheShapeHash: 'BA27D965',
    planCacheKey: '7A892B81',
    optimizationTimeMillis: 5,
    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' }
    },
```

Выполнение запроса без индекса:

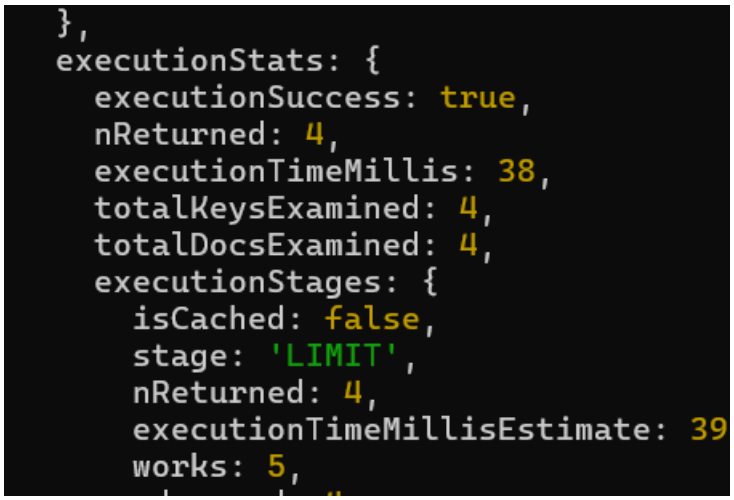


executionTimeMillis = 157

Создайте индекс для ключа value.



Выполнение запроса с индексом:



executionTimeMillis = 38

Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

Запрос с использованием индекса выполняется быстрее.

Вывод:

Я овладела практическими навыками работы с СУБД MongoDB.