Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Мобильные и сетевые технологии**

Направление подготовки **09.03.03 Прикладная информатика**

О Т Ч Е Т

Лабораторная работа 6

Тема: Введение. Работа с БД в СУБД MongoDB.

Обучающийся: Файзулин Радмир Русланович, группы К3239

Преподаватель: Говорова Марина Михайловна

Санкт-Петербург 2024

СОДЕРЖАНИЕ

Цель работы:

Овладеть практическими навыками установки СУБД MongoDB. 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.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 6.0.6 (текущая)

Практическое задание:

1. Установите MongoDB для обеих типов систем (32/64 бита).
2. Проверьте работоспособность системы запуском клиента mongo.
3. Выполните методы:
    a) db.help()
    b) db.help
    c) db.stats()
4. Создайте БД learn.
5. Получите список доступных БД.
6. Создайте коллекцию unicorns, вставив в нее документ `{name: 'Aurora', gender: 'f', weight: 450}`.
7. Просмотрите список текущих коллекций.
8. Переименуйте коллекцию unicorns.
9. Просмотрите статистику коллекции.
10. Удалите коллекцию.
11. Удалите БД learn.

Ход работы

Выполнение практического задания 6.1

3
db.help()

```
test> db.help()

  Database Class:

    getMongo                                  Returns the current datal
    getName                                   Returns the name of the
    getCollectionNames                        Returns an array contain
atabase.
    getCollectionInfos                        Returns an array of docu
```

db.help

```
test> db.help

  Database Class:

    getMongo                                  Returns the current database co
    getName                                   Returns the name of the DB
    getCollectionNames                        Returns an array containing the
atabase.
```

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 Создание learn

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

5 Список бд

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

6 Добавление в unicorns

```
learn> db.unicorns.insertOne({name: 'Aurora', gender: 'f', weight: 450})
{
  acknowledged: true,
  insertedId: ObjectId('660e92cabda2657e119f990a')
}
```

7 Список коллекций

```
learn> show collections
unicorns
```

8 rename

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

9 stats

```
learn> db.pets.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_times
tamp=none,write_timestamp=on),block_allocation=best,block_compressor=snappy,cach
e_resident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0,encrypti
```

10 Удаление коллекции

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

11 Удаление БД learn

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

Выполнение практического задания 6.2

2.1.1

3) Добавление вторым способом

```
learn> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gen
der: 'm', vampires: 165})
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660e9742bda2657e119f9916') }
}
learn>
```

4) Проверка содержимого с find

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('660e964ebda2657e119f990b'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600
```

2.2.1

1)

Вывод самцов

```
learn> db.unicorns.find({gender: "m"})
[
```

Вывод самок

```
learn> db.unicorns.find({gender: "f"})
```

Ограничение до трёх самок

```
learn> db.unicorns.find({gender: "f"}).limit(3)
```

Сорт по имени

```
learn> db.unicorns.find({gender: "f"}).sort({name: 1})
[
  {
```

```
learn> db.unicorns.find({gender: "m"}).sort({name: 1})
```

2)

Самка любящая морковь: (findOne)

```
learn> db.unicorns.findOne({gender: "f", loves: "carrot"})
{
  _id: ObjectId('660e964ebda2657e119f990c'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```

1

```
learn> db.unicorns.find({gender: "f", loves: "carrot"}).limit(1)
[
  {
    _id: ObjectId('660e964ebda2657e119f990c'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

2.2.2

Список самцов, исключив инфу о предпочтениях и поле

```
learn> db.unicorns.find({gender: "m"}, {gender: 0, loves: 0})
[
  {
    _id: ObjectId('660e964ebda2657e119f990b'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
```

2.2.3

Единороги в обратном порядке добавления

```
learn> db.unicorns.find().sort({ $natural: -1})
[
  {
    _id: ObjectId('660e9742bda2657e119f9916'),
    name: 'Dunx',
```

2.1.4

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

```
learn> db.unicorns.find({}, {_id: 0, loves: {$slice :1}})
[
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
```

2.3.1

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

```
learn> db.unicorns.find({weight: {$gte: 500, $lte: 700}}, {_id: 0})
[
  {
```

2.3.2

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

```
learn> db.unicorns.find({weight: {$gte: 500, $lte: 700}, loves: {$all: ['gra
pe', 'lemon']}}, {_id: 0})
[
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

2.3.3

Найти всех единорогов, не имеющих ключ `vampires`.

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('660e964ebda2657e119f9915'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn>
```

2.3.4

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

```
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

1) *Создайте коллекцию towns, включающую следующие документы:*

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660ea724bda2657e119f9917') }
}
learn> db.towns.insert({name: "New York", populatiuon: 22200000, last_sensus
learn> db.towns.insert({name: "New York", populatiuon: 22200000, last_sensus
: ISODate("2009-07-31"), famous_for: ["status of liberty", "food"], mayor: {
name: "Michael Bloomberg", party: "I"}})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660ea7aebda2657e119f9918') }
}
learn> db.towns.insert({name: "Portland", populatiuon: 528000, last_sensus:
ISODate("2009-07-20"), famous_for: ["beer", "food"], mayor: {name: "Sam Adam
s", party: "D"}})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660ea7c3bda2657e119f9919') }
}
```

2) *Сформировать запрос, который возвращает список городов с независимыми мэрами (*`party="I"`*). Вывести 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' }
  }
]
learn>
```

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

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

3.1.2

3) *Сформировать функцию для вывода списка самцов единорогов.*

```
learn> function fn() {return db.unicorns.find({gender: "m"});}
[Function: fn]
learn> fn()
[
  {
    _id: ObjectId('660e964ebda2657e119f990b'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
```

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

```
learn> cursor.sort({name: -1}).limit(2);
[
  {
    _id: ObjectId('660e964ebda2657e119f990d'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('660e964ebda2657e119f990e'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  }
]
```

5) *Вывести результат, используя forEach.*

```
learn> var cursor = fn(); null;
null
learn> cursor.sort({name: -1}).limit(2); null;
null
learn> cursor.forEach(function(obj){print(obj._id, obj.name, obj.loves, obj.
weight, obj.gender, obj.vampires);})
ObjectId('660e964ebda2657e119f990d') Unicrom [ 'energon', 'redbull' ] 984 m
182
ObjectId('660e964ebda2657e119f990e') Roooooodles [ 'apple' ] 575 m 99

learn>
```

6) *Содержание коллекции единорогов* `unicorns`*:*

done ;)

3.2.1

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

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 обоих полов.

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

3.3.1

1. *Выполнить команду:*

```
> db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
```

Проверить содержимое коллекции `unicorns`.

Я так понял метод save в новых версиях не поддерживается. (позже понял, что аналог - это update+upsert)

```
learn> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340,
gender: 'm'})
{
  acknowledged: true,
  insertedId: ObjectId('660eefd0bda2657e119f991a')
}
learn> db.unicorns.find()
[
    _id: ObjectId('660eefd0bda2657e119f991a'),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
learn>
```

3.3.2

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

2. Проверить содержимое коллекции `unicorns`*.*

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

```
 },
 {
   _id: ObjectId('660e964ebda2657e119f9910'),
   name: 'Ayna',
   loves: [ 'strawberry', 'lemon' ],
   weight: 800,
   gender: 'f',
   vampires: 51
 },
 {
```

3.3.3

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

Проверить содержимое коллекции `unicorns`*.*

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

```
{
  _id: ObjectId('660e964ebda2657e119f9912'),
  name: 'Raleigh',
  loves: 'redbull',
  weight: 421,
  gender: 'm',
  vampires: 2
},
```

3.3.4

Всем самцам единорогов увеличить количество убитых вапмиров на 5.

Проверить содержимое коллекции `unicorns`*.*

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

3.3.5

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

Проверить содержимое коллекции `towns`.

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

```
    _id: ObjectId('660ea7c3bda2657e119f9919'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

3.3.6

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

Проверить содержимое коллекции `unicorns`.

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

```
  _id: ObjectId('660e964ebda2657e119f9914'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon', 'chocolate' ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
```

3.3.7

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

Проверить содержимое коллекции `unicorns`*.*

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

```
    _id: ObjectId('660e964ebda2657e119f990c'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
```

3.4.1

4) *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"}}
```

5) *Удалите документы с беспартийными мэрами.*

```
learn> db.towns.remove({"mayor.party": {$exists: false}})
DeprecationWarning: Collection.remove() is deprecated. Use delete
One, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId('660efaaebda2657e119f991e'),
    name: 'New York',
```

6) *Проверьте содержание коллекции.*

```
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId('660efaaebda2657e119f991e'),
    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('660efabdbda2657e119f991f'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
learn>
```

7) *Очистите коллекцию.*

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

8) *Просмотрите список доступных коллекций.*

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

4.1.1

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

```
learn> db.countries.insert({_id: "ru", name: "russia", description: "big"})
{ acknowledged: true, insertedIds: { '0': 'ru' } }
learn> db.countries.insert({_id: "ca", name: "canada", description: "big"})
{ acknowledged: true, insertedIds: { '0': 'ca' } }
learn> db.countries.insert({_id: "fr", name: "france", description: "medium"})
{ acknowledged: true, insertedIds: { '0': 'fr' } }
learn> db.countries.insert({_id: "ir", name: "ireland", description: "small"})
{ acknowledged: true, insertedIds: { '0': 'ir' } }
learn>
```

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

```
learn> db.unicorns.update({_id: ObjectId('660e964ebda2657e119f990b')}, {$set: {country:{$ref: "countries", $_id: "
ru"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.update({_id: ObjectId('660e964ebda2657e119f990c')}, {$set: {country:{$ref: "countries", $_id: "
ca"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.update({_id: ObjectId('660e964ebda2657e119f990d')}, {$set: {country:{$ref: "countries", $_id: "
ir"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find()
```

9)*Проверьте содержание коллекции единорогов.*

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('660e964ebda2657e119f990b'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    country: { '$ref': 'countries', '$_id': 'ru' }
  },
  {
    _id: ObjectId('660e964ebda2657e119f990c'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    country: { '$ref': 'countries', '$_id': 'ca' }
  },
  {
    _id: ObjectId('660e964ebda2657e119f990d'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187,
    country: { '$ref': 'countries', '$_id': 'ir' }
```

10)*Содержание коллекции единорогов* `unicorns`*:*
done ;)

4.2.1

1)*Проверьте, можно ли задать для коллекции* `unicorns` *индекс для ключа* `name` *с флагом* `unique`

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

Видимо удалось.
2)Содержание коллекции единорогов unicorns:
done ;)
4.3.1
1) Получите информацию о всех индексах коллекции unicorns .

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

2) Удалите все индексы, кроме индекса для идентификатора.

```
learn> db.unicorns.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn> db.unicorns.createIndex({ "name": 1 }, { unique: true })
name_1
learn> db.unicorns.dropIndex()
MongoshInvalidInputError: [COMMON-10001] Missing required argument at position 0 (Collection.dropIndex)
learn> db.unicorns.dropIndex("name")
MongoshInternalError[IndexNotFound]: index not found with name [name]
learn> db.unicorns.dropIndex("name_1")
{ nIndexesWas: 2, ok: 1 }
learn> db.unicorns.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn>
```

3) Попытайтесь удалить индекс для идентификатора.

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

4.4.1

1)*Создайте объемную коллекцию* `numbers`*, задействовав курсор:*

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

2)*Выберите последних четыре документа.*

```
learn> db.numbers.find().skip(99996)
[
  { _id: ObjectId('660f1dcfbda2657e11a11fd4'), value: 99996 },
  { _id: ObjectId('660f1dcfbda2657e11a11fd5'), value: 99997 },
  { _id: ObjectId('660f1dcfbda2657e11a11fd6'), value: 99998 },
  { _id: ObjectId('660f1dcfbda2657e11a11fd7'), value: 99999 }
]
```

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

```
executionSuccess: true,
nReturned: 4,
executionTimeMillis: 47,
totalKeysExamined: 0,
totalDocsExamined: 100000,
```

(47)

4)*Создайте индекс для ключа* `value`*.*

```
learn> db.numbers.ensureIndex({"values": 1})
[ 'values_1' ]
```

5)*Получите информацию о всех индексах коллекции* `numbres`.

```
learn> db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { values: 1 }, name: 'values_1' }
]
learn>
```

6)*Выполните запрос 2.*

```
learn> db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { values: 1 }, name: 'values_1' }
]
learn> db.numbers.find().skip(99996)
[
  { _id: ObjectId('660f1dcfbda2657e11a11fd4'), value: 99996 },
  { _id: ObjectId('660f1dcfbda2657e11a11fd5'), value: 99997 },
```

7)*Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?*

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 47,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
  executionStages: {
    stage: 'SKIP',
```

Столько же(???)

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

По моим результатам эффективность по времени в обоих случаях оказалась одинаковая. Возможно, для опыта стоило использовать не skip, и тогда у запроса с индексом был шанс обойти запрос с его отсутствием ;)

Дополнительное индивидуальное задание

insert (one, many), update, delete. Синтаксис и примеры.

пример использования insertOne

```
learn> db.users.insertOne({"name": "Radmir", "age": 27, "languages": "tatarian"})
{
  acknowledged: true,
  insertedId: ObjectId('660f2691bda2657e11a11fda')
}
learn>
```

пример использования insertMany

```
learn> db.users.insertMany([{"name": "Tom", "age": 28, "languages": "spanish"}, {"name": "Bill", "age": 32, "languages": "french"}]
)
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('660f2635bda2657e11a11fd8'),
    '1': ObjectId('660f2635bda2657e11a11fd9')
  }
}
```

Как видно по названию, отличаются они количеством добавляемых документов в коллекцию. В случае One это ровно один, в случае Many уже несколько+

Какими способами можно добавить данные (одну запись или несколько) в массив. Чем они отличаются? Приведите два способа.

В массив можно добавить с помощью команд $push и $addToSet.
В первом случае данные добавятся только к уже существующем и ТОЛЬКО ОДИН ЛОТ, а во втором даже если их не было до этого. Более того, с помощью $each команда $addToSet способна добавить сразу НЕСКОЛЬКО ЛОТОВ в массив. Ниже примеры каждого из операторов:

```
learn> db.users.insert({"name": "Tom"},  {$addToSet: {languages: {$each: ["bashkirskiy", "tatarskiy"]}}})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f28b7bda2657e11a11fdb') }
}
learn>
```

```
learn> db.users.update({"name": "Tom"},  {$push: {"languages": "estonian"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Создайте новую базу данных MongoDB с именем "task11db". В этой базе данных создайте две коллекции: "students" и "courses". Добавьте несколько записей в каждую коллекцию.

```
learn> use task11db
task11db>
```

```
task11db> db.createCollection("students")
{ ok: 1 }
task11db> db.createCollection("courses")
{ ok: 1 }
task11db>
```

```
task11db> db.courses.find()
[
  { _id: ObjectId('660f3c6bbda2657e11a11fe8'), name: 'first', age: 18 },
  { _id: ObjectId('660f3c92bda2657e11a11fe9'), name: 'sirst', age: 19 }
]
task11db>
```

Используйте DBRef для связывания студентов с курсами, в которых они учатся.

Добавление ссылки:

```
task11db> radmir = ({"name": "Radmir", "group": 3239, cours: db.courses.findOne()._id})
{
  name: 'Radmir',
  group: 3239,
  cours: ObjectId('660f3c6bbda2657e11a11fe8')
}
```

```
task11db> db.students.insert(radmir)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('660f3e6abda2657e11a11fea') }
}
task11db>
```

Коллекция students после добавления ссылки

```
task11db> db.students.find()
[
  {
    _id: ObjectId('660f3e6abda2657e11a11fea'),
    name: 'Radmir',
    group: 3239,
    cours: ObjectId('660f3c6bbda2657e11a11fe8')
  }
]
```

Напишите запросы, чтобы получить информацию о студентах и их курсах, используя связи DBRef.

```
task11db> db.courses.findOne({_id: radmir.cours})
{ _id: ObjectId('660f3c6bbda2657e11a11fe8'), name: 'first', age: 18 }
task11db>
```

Вывод

Я овладел практическими навыками установки СУБД MongoDB. Овладел 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. Познакомился с MongoDB с различных сторон, затем выполнил индивидуальное практическое задание.