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 (*Березина Софья Константиновна)*
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
2024/2025

ОГЛАВЛЕНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель работы: 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.

2. ПРАКТИЧЕСКОЕ ЗАДАНИЕ

1. Часть 1

1.1. Установите MongoDB для обеих типов систем (32/64 бита).

1.2. Проверьте работоспособность системы запуском клиента mongo.

1.3. Выполните методы:

1.3.1. db.help()

1.3.2. db.help

1.3.3. db.stats()

1.4. Создайте БД learn.

1.5. Получите список доступных БД.

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

1.7. Просмотрите список текущих коллекций.

1.8. Переименуйте коллекцию unicorns.

1.9. Просмотрите статистику коллекции.

1.10. Удалите коллекцию.

1.11. Удалите БД learn.

2. Часть 2

2.1. CRUD-операции в СУБД MongoDB

2.2. Запросы к БД MongoDB

2.3. Ссылки в БД

3. ВЫПОЛНЕНИЕ РАБОТЫ. ЧАСТЬ 1

3.1 Установка MongoDB и выполнение методов

Выполнение 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
```

Выполнение 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 variabl
e in the shell environment.
    getCollection                             Returns a collection or a view object that is functionall
```

Выполнение метода 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
}
```

3.2 Работа с БД learn

Создаем БД learn:

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

Создаем коллекцию unicorns с документом:

```
learn> db.unicorns.insertOne({
...   name: 'Aurora',
...   gender: 'f',
...   weight: 450
... })
{
  acknowledged: true,
  insertedId: ObjectId('6832f2d61ba517b298379396')
}
learn>
```

Просматриваем список коллекций:

```
learn> show collections
unicorns
```

Переименуем коллекцию:

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

Посмотрим статистику:

```
learn> db.horses.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_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=(compare_timestamp=oldest_timestamp,enabled=false,file_m
etadata=,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_pag
e_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,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,prefix_compression=false,prefix_compression_min=4,sour
ce=,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-7-1697945366446011974',
    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,
```

Удалим коллекцию:

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

learn>
```

Удалим БД:

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

4. ВЫПОЛНЕНИЕ РАБОТЫ. ЧАСТЬ 2

4.1 CRUD-операции в СУБД MongoDB

1. *Создайте базу данных learn.*
2. *Заполните коллекцию единорогов unicorns:*

```
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'});
```

3. *Используя второй способ, вставьте в коллекцию единорогов документ:*

```
{name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
```

4. *Проверьте содержимое коллекции с помощью метода find.*

```
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', vampir
es: 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', vampire
s: 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('6832f6081ba517b2983793a1') }
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('6832f6081ba517b298379397'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6832f6081ba517b298379398'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
```

```
learn> document = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 16
5});
... db.unicorns.insert(document);
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6832f62e1ba517b2983793a2') }
}
```

```
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
}
```

```
    _id: ObjectId('6832f6081ba517b2983793a1'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6832f62e1ba517b2983793a2'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
```

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

Список самцов:

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('6832f62e1ba517b2983793a2'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6832f6081ba517b298379397'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6832f6081ba517b29837939d'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6832f6081ba517b2983793a0'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
```

```
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6832f6081ba517b29837939e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('6832f6081ba517b29837939a'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6832f6081ba517b298379399'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

Список самок:

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1})
[
  {
    _id: ObjectId('6832f6081ba517b298379398'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6832f6081ba517b29837939c'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('6832f6081ba517b29837939f'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
```

```
    vampires: 33
  },
  {
    _id: ObjectId('6832f6081ba517b2983793a1'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6832f6081ba517b29837939b'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
]
```

Список всех самок, которые любят carrot:

```
learn> db.unicorns.findOne({
...   gender: 'f',
...   loves: 'carrot'
... })
{
  _id: ObjectId('6832f6081ba517b298379398'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}


learn> db.unicorns.find({
...   gender: 'f',
...   loves: 'carrot'
... }).limit(1)
[
  {
    _id: ObjectId('6832f6081ba517b298379398'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

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

```
learn> db.unicorns.find(
```

```
...   {gender: 'm'},
...   {
...     loves: 0,
...     gender: 0,
...     _id: 0
...   }
... ).sort({name: 1})
[
  { name: 'Dunx', weight: 704, vampires: 165 },
  { name: 'Horny', weight: 600, vampires: 63 },
  { name: 'Kenny', weight: 690, vampires: 39 },
  { name: 'Pilot', weight: 650, vampires: 54 },
  { name: 'Raleigh', weight: 421, vampires: 2 },
  { name: 'Roooooodles', weight: 575, vampires: 99 },
  { name: 'Unicrom', weight: 984, vampires: 182 }
]
```

8. *Вывести список единорогов в обратном порядке добавления.*

```
learn> db.unicorns.find().sort({_id: -1})
[
  {
    _id: ObjectId('6832f62e1ba517b2983793a2'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6832f6081ba517b2983793a1'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6832f6081ba517b2983793a0'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6832f6081ba517b29837939f'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('6832f6081ba517b29837939e'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
```

```
  gender: 'm',
  vampires: 2
},
{
  _id: ObjectId('6832f6081ba517b29837939d'),
  name: 'Kenny',
  loves: [ 'grape', 'lemon' ],
  weight: 690,
  gender: 'm',
  vampires: 39
},
{
  _id: ObjectId('6832f6081ba517b29837939c'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 733,
  gender: 'f',
  vampires: 40
},
{
  _id: ObjectId('6832f6081ba517b29837939b'),
  name: 'Solnara',
  loves: [ 'apple', 'carrot', 'chocolate' ],
  weight: 550,
  gender: 'f',
  vampires: 80
},
{
  _id: ObjectId('6832f6081ba517b29837939a'),
  name: 'Roooooodles',
  loves: [ 'apple' ],
  weight: 575,
  gender: 'm',
  vampires: 99
},
{
  _id: ObjectId('6832f6081ba517b298379399'),
  name: 'Unicrom',
  loves: [ 'energon', 'redbull' ],
  weight: 984,
  gender: 'm',
  vampires: 182
},
{
  _id: ObjectId('6832f6081ba517b298379398'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
  _id: ObjectId('6832f6081ba517b298379397'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
```

```
    gender: 'm',
    vampires: 63
  }
]
```

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

```
learn> db.unicorns.aggregate([
...   {
...     $project: {
...       _id: 0,
...       name: 1,
...       firstLove: {$arrayElemAt: ["$loves", 0]},
...       weight: 1,
...       gender: 1,
...       vampires: 1
...     }
...   }
... ])
[
  {
    name: 'Horny',
    weight: 600,
    gender: 'm',
    vampires: 63,
    firstLove: 'carrot'
  },
  {
    name: 'Aurora',
    weight: 450,
    gender: 'f',
    vampires: 43,
    firstLove: 'carrot'
  },
  {
    name: 'Unicrom',
    weight: 984,
    gender: 'm',
    vampires: 182,
    firstLove: 'energon'
  },
  {
    name: 'Roooooodles',
    weight: 575,
    gender: 'm',
    vampires: 99,
    firstLove: 'apple'
  },
  {
    name: 'Solnara',
    weight: 550,
    gender: 'f',
    vampires: 80,
    firstLove: 'apple'
  },
  {
```

```
    name: 'Ayna',
    weight: 733,
    gender: 'f',
    vampires: 40,
    firstLove: 'strawberry'
  },
  {
    name: 'Kenny',
    weight: 690,
    gender: 'm',
    vampires: 39,
    firstLove: 'grape'
  },
  {
    name: 'Raleigh',
    weight: 421,
    gender: 'm',
    vampires: 2,
    firstLove: 'apple'
  },
  {
    name: 'Leia',
    weight: 601,
    gender: 'f',
    vampires: 33,
    firstLove: 'apple'
  },
  {
    name: 'Pilot',
    weight: 650,
    gender: 'm',
    vampires: 54,
    firstLove: 'apple'
  },
  { name: 'Nimue', weight: 540, gender: 'f', firstLove: 'grape' },
  {
    name: 'Dunx',
    weight: 704,
    gender: 'm',
    vampires: 165,
    firstLove: 'grape'
  }
]
```

10. *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'
  }
]
```

11. *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
  }
]
```

12. *Найти всех единорогов, не имеющих ключ* `vampires`.

```
learn> db.unicorns.find({
...   vampires: {$exists: false}
... })
[
  {
    _id: ObjectId('6832f6081ba517b2983793a1'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

13. *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,
```

```
...     firstPreference: {$arrayElemAt: ["$loves", 0]}
...   }},
...   {$sort: {name: 1}}
... ])
[
  { name: 'Dunx', firstPreference: 'grape' },
  { name: 'Horny', firstPreference: 'carrot' },
  { name: 'Kenny', firstPreference: 'grape' },
  { name: 'Pilot', firstPreference: 'apple' },
  { name: 'Raleigh', firstPreference: 'apple' },
  { name: 'Roooooodles', firstPreference: 'apple' },
  { name: 'Unicrom', firstPreference: 'energon' }
]
```

4.2 Запросы к БД MongoDB

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"}}
```

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' }
  }

]
```

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, mayor: 1}
... )
[ { name: 'Punxsutawney', mayor: { name: 'Jim Wehrle' } } ]
```

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

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

4. *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);
```

5. *Вывести результат, используя forEach.*

```
learn> cursor.forEach(function(unicorn) {
...   printjson(unicorn);
... });
{
  _id: ObjectId('6832f62e1ba517b2983793a2'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('6832f6081ba517b298379397'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

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

```
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', 44), 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'});

db.unicorns.insert ({name: 'Dunx', loves: ['grape', 'watermelon'], weight:
704, gender: 'm', vampires: 165})
```

7. *Вывести количество самок единорогов весом от полутонны до 600 кг.*

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

```
4
```

8. *Вывести список предпочтений.*

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

9. *Посчитать количество особей единорогов обоих полов.*

```
learn> db.unicorns.aggregate([
...   {$group: {
```

```
...     _id: "$gender",
...     count: {$sum: 1}
...   }}
... ])

[ { _id: 'f', count: 10 }, { _id: 'm', count: 14 } ]
```

10. *Выполнить команду:*

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

```
learn> db.unicorns.insertOne({
...   name: 'Barny',
...   loves: ['grape'],
...   weight: 340,
...   gender: 'm'
... })
{
  acknowledged: true,
  insertedId: ObjectId('68331b811ba517b2983793bf')
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('68331b571ba517b2983793b3'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68331b571ba517b2983793b4'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('68331b571ba517b2983793b5'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('68331b571ba517b2983793b6'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
```

```
  vampires: 99
},
{
  _id: ObjectId('68331b571ba517b2983793b7'),
  name: 'Solnara',
  loves: [ 'apple', 'carrot', 'chocolate' ],
  weight: 550,
  gender: 'f',
  vampires: 80
},
{
  _id: ObjectId('68331b571ba517b2983793b8'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 733,
  gender: 'f',
  vampires: 40
},
{
  _id: ObjectId('68331b571ba517b2983793b9'),
  name: 'Kenny',
  loves: [ 'grape', 'lemon' ],
  weight: 690,
  gender: 'm',
  vampires: 39
},
{
  _id: ObjectId('68331b571ba517b2983793ba'),
  name: 'Raleigh',
  loves: [ 'apple', 'sugar' ],
  weight: 421,
  gender: 'm',
  vampires: 2
},
{
  _id: ObjectId('68331b571ba517b2983793bb'),
  name: 'Leia',
  loves: [ 'apple', 'watermelon' ],
  weight: 601,
  gender: 'f',
  vampires: 33
},
{
  _id: ObjectId('68331b571ba517b2983793bc'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon' ],
  weight: 650,
  gender: 'm',
  vampires: 54
},
{
  _id: ObjectId('68331b571ba517b2983793bd'),
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f'
```

```
},
{
  _id: ObjectId('68331b571ba517b2983793be'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
},
{
  _id: ObjectId('68331b811ba517b2983793bf'),
  name: 'Barny',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm'
}
   ]
```

11. *Для самки единорога* `Ayna` *внести изменения в БД: теперь ее вес 800, она убила 51 вапмира. Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.updateOne(
...   {name: 'Ayna'},
...   {$set: {weight: 800, vampires: 51}}
... )
learn> db.unicorns.findOne(
...   {name: 'Ayna'},
...   {_id: 0, name: 1, weight: 1, vampires: 1}
... )
{ name: 'Ayna', weight: 800, vampires: 51 }
```

12. *Для самца единорога* `Raleigh` *внести изменения в БД: теперь он любит рэдбул. Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.updateOne(
...   {name: 'Raleigh'},
...   {$push: {loves: 'redbull'}}
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.findOne(
...   {name: 'Raleigh'},
...   {_id: 0, name: 1, loves: 1}
... )
{ name: 'Raleigh', loves: [ 'apple', 'sugar', 'redbull' ] }
```

13. *Всем самцам единорогов увеличить количество убитых вапмиров на 5. Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.updateMany(
...   {gender: 'm', vampires: {$exists: true}},
...   {$inc: {vampires: 5}}
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
learn> db.unicorns.find(
...   {gender: 'm', vampires: {$exists: true}},
...   {_id: 0, name: 1, vampires: 1}
... ).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 }
]
```

14. *Изменить информацию о городе Портланд: мэр этого города теперь беспартийный. Проверить содержимое коллекции* `towns`.

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

15. *Изменить информацию о самце единорога Pilot: теперь он любит и шоколад. Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.updateOne(
...   {name: "Pilot"},
...   {$push: {loves: "chocolate"}}
... )
{
```

```
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.findOne({name: "Pilot"}, {_id: 0, name: 1, loves: 1})
{ name: 'Pilot', loves: [ 'apple', 'watermelon', 'chocolate' ] }
```

16. *Изменить информацию о самке единорога* `Aurora`: *теперь она любит еще и сахар, и лимоны. Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.updateOne(
...   {name: "Aurora"},
...   {$push: {loves: {$each: ["sugar", "lemon"]}}}
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.findOne({name: "Aurora"}, {_id: 0, name: 1, loves:
1})
{ name: 'Aurora', loves: [ 'carrot', 'grape', 'sugar', 'lemon' ] }
```

17. *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. Проверьте содержание коллекции.

```
learn> db.towns.deleteMany({"mayor.party": {$exists: false}})
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId('68331e931ba517b2983793c1'),
    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('68331e931ba517b2983793c2'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }

    ]
```

18. *Очистите коллекцию. Просмотрите список доступных коллекций.*

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

unicorns
```

4.3 Ссылки в БД

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

```
learn> db.habitats.insertMany([
...   {
...     _id: "forest",
...     fullName: "Волшебный лес",
...     description: "Густой лес с магическими растениями"
...   },
...   {
...     _id: "mountains",
...     fullName: "Хрустальные горы",
...     description: "Высокие горы с кристальными пещерами"
...   },
...   {
...     _id: "meadow",
...     fullName: "Радужные луга",
...     description: "Цветущие луга с радужными цветами"
...   }
... ])
```

```
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'mountains', '2': 'meadow' }
}
```

2. *Включите для нескольких единорогов в документы ссылку на зону обитания, использую второй способ автоматического связывания. Проверьте содержание коллекции едиорогов.*

```
learn> db.unicorns.updateMany(
...   {name: {$in: ["Horny", "Aurora", "Unicrom"]}},
...   {
...     $set: {
...       habitat: new DBRef("habitats", "forest", "learn")
...     }
...   }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
learn> db.unicorns.find(
...   {habitat: {$exists: true}},
...   {name: 1, habitat: 1, _id: 0}
... )
[
  { name: 'Horny', habitat: DBRef('habitats', 'forest', 'learn') },
  { name: 'Aurora', habitat: DBRef('habitats', 'forest', 'learn')
},
  { name: 'Unicrom', habitat: DBRef('habitats', 'forest', 'learn')
}

]
```

3. *Содержание коллекции единорогов* `unicorns`*:*

```
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', 44), 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'});

db.unicorns.insert {name: 'Dunx', loves: ['grape', 'watermelon'], weight:
704, gender: 'm', vampires: 165}
```

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

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

[ 'name_1' ]
```

5. *Содержание коллекции единорогов* `unicorns`*:*

```
db.unicorns.insert({name: 'Horny', dob: new Date(1992,2,13,7,47), loves:
['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});

db.unicorns.insert({name: 'Aurora', dob: new Date(1991, 0, 24, 13, 0), loves:
['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', dob: new Date(1973, 1, 9, 22, 10), loves:
['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});

db.unicorns.insert({name: 'Roooooodles', dob: new Date(1979, 7, 18, 18, 44),
loves: ['apple'], weight: 575, gender: 'm', vampires: 99});

db.unicorns.insert({name: 'Solnara', dob: new Date(1985, 6, 4, 2, 1),
loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f',
vampires:80});

db.unicorns.insert({name:'Ayna', dob: new Date(1998, 2, 7, 8, 30), loves:
['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});

db.unicorns.insert({name:'Kenny', dob: new Date(1997, 6, 1, 10, 42), loves:
['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});

db.unicorns.insert({name: 'Raleigh', dob: new Date(2005, 4, 3, 0, 57), loves:
['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});

db.unicorns.insert({name: 'Leia', dob: new Date(2001, 9, 8, 14, 53), loves:
['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});

db.unicorns.insert({name: 'Pilot', dob: new Date(1997, 2, 1, 5, 3), loves:
['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});

db.unicorns.insert ({name: 'Nimue', dob: new Date(1999, 11, 20, 16, 15),
loves: ['grape', 'carrot'], weight: 540, gender: 'f'});

db.unicorns.insert ({name: 'Dunx', dob: new Date(1976, 6, 18, 18, 18), loves:
['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165});
```

6. *Получите 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 }
]
```

7. *Удалите все индексы, кроме индекса для идентификатора.*

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

8. *Попытайтесь удалить индекс для идентификатора.*

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

9. *Создайте объемную коллекцию* `numbers`*, задействовав курсор:*

`for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}` *Выберите последних четыре документа.*

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

{
  acknowledged: true,
  insertedId: ObjectId('683456d31ba517b298391a7b')
}
learn>

learn> db.numbers.find().sort({_id: -1}).limit(4)
[
  { _id: ObjectId('683456d31ba517b298391a7b'), value: 99999 },
  { _id: ObjectId('683456d31ba517b298391a7a'), value: 99998 },
  { _id: ObjectId('683456d31ba517b298391a79'), value: 99997 },
  { _id: ObjectId('683456d31ba517b298391a78'), value: 99996 }
        ]
```

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

```
learn> db.numbers.find().sort({_id: -
1}).limit(4).explain("executionStats").executionStats.executionTime
Millis
0
```

11. *Создайте индекс для ключа* `value`*. Получите информацию о всех индексах коллекции* `numbres`*.*

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

```
]
```

12. Выполните запрос 2.

```
learn> db.numbers.find().sort({value: -1}).limit(4)
[
  { _id: ObjectId('683456d31ba517b298391a7b'), value: 99999 },
  { _id: ObjectId('683456d31ba517b298391a7a'), value: 99998 },
  { _id: ObjectId('683456d31ba517b298391a79'), value: 99997 },
  { _id: ObjectId('683456d31ba517b298391a78'), value: 99996 }
]
```

13. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?

```
learn> db.numbers.find().sort({value: -
1}).limit(4).explain("executionStats").executionStats.executionTime
Millis
0
```

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

В примере не удалось на практике убедиться в эффективности индексов, но в теории они сильно ускоряют работу, особенно при работе с большими данными.

5. ВЫВОДЫ

В ходе выполнения лабораторной работы были успешно освоены ключевые аспекты работы с MongoDB, включая создание и модификацию коллекций, выполнение запросов различной сложности и оптимизацию производительности с использованием индексов. Полученные навыки позволяют эффективно работать с MongoDB, оптимизировать запросы и понимать принципы хранения данных в документоориентированных СУБД. Лабораторная работа продемонстрировала важность правильного проектирования коллекций и использования индексов для обеспечения высокой производительности при работе с большими объемами данных.