Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №6 «ВВЕДЕНИЕ В СУБД MONGODB. УСТАНОВКА MONGODB. НАЧАЛО РАБОТЫ С БД»

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Трубников А.П

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова М.М.

Санкт-Петербург 2024

Цель работы:

овладеть практическими навыками установки СУБД MongoDB..

Практическое задание:

1. Установите MongoDB для обеих типов систем (32/64 бита).
2. Проверьте работоспособность системы запуском клиента mongo.
3. Выполните методы:

db.help()

db.help

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.

Выполнение практических заданий.

Задание 3

Команда db.help()

```
> 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 error if the user already
                            exists on the database.
  updateUser                Updates the user's profile on the database on which you run the method. An update to a field completely replaces the previous
                            field's 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 created the user.
  logout                    Ends the current authentication session. This function has no effect if the current session is not authenticated.
  dropUser                  Removes the user from the current database.
  dropAllUsers              Removes all users from the current database.
  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.
```

Команда db.help

```
>_MONGOSH
> 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 error if the user already
                            exists on the database.
  updateUser                Updates the user's profile on the database on which you run the method. An update to a field completely replaces the previous
                            field's 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 created the user.
  logout                    Ends the current authentication session. This function has no effect if the current session is not authenticated.
  dropUser                  Removes the user from the current database.
  dropAllUsers              Removes all users from the current database.
  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.
```

Команда db.stats()

```
> db.stats()
< {
    db: 'test',
    collections: Long('0'),
    views: Long('0'),
    objects: Long('0'),
    avgObjSize: 0,
    dataSize: 0,
    storageSize: 0,
    indexes: Long('0'),
    indexSize: 0,
    totalSize: 0,
    scaleFactor: Long('1'),
    fsUsedSize: 0,
    fsTotalSize: 0,
    ok: 1
  }
test>
```

Задание 4

Создание базы данных:

```
> use <Lab6_BD>
< switched to db <Lab6_BD>
<Lab6_BD>>
```

Задание 5

Получите список доступных БД.:

```
> show dbs
< Lab6_BD  40.00 KiB
  admin    40.00 KiB
  config   96.00 KiB
  local    40.00 KiB
```

Задание 6:

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

```
> db.createCollection("unicorns")
< { ok: 1 }
> db.unicorns.insertOne({name: 'Aurora', gender: 'f', weight: 450})
< {
    acknowledged: true,
    insertedId: ObjectId('65e1987c8cb7b7b6bbcdaffe')
  }
```

Задание 7:

Просмотрите список текущих коллекций.

```
> show collections
< unicorns
```

Задание 8:

Переименуйте коллекцию unicorns.

```
> db.unicorns.renameCollection("pages")
< { ok: 1 }
```

Задание 9:

Просмотрите статистику коллекции:

```
> db.pages.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_
      type: 'file',
      uri: 'statistics:table:collection-7-4160459546396688383',
      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,
```

```
    }
  },
  sharded: false,
  size: 65,
  count: 1,
  numOrphanDocs: 0,
  storageSize: 20480,
  totalIndexSize: 20480,
  totalSize: 40960,
  indexSizes: { _id_: 20480 },
  avgObjSize: 65,
  ns: 'Lab6_BD.pages',
  nindexes: 1,
  scaleFactor: 1
```

Задание 10:

Удалите коллекцию

```
> db.pages.drop()
< true
```

Задание 11:

Удалите БД learn.

```
> db.dropDatabase()
< { ok: 1, dropped: 'Lab6_BD' }
```

Практическое задание 2.1.1:

Вывод:

В ходе выполнения практического задания по установке СУБД MongoDB были изучены основные шаги по установке и настройке сервера базы данных.

```
> db.createCollection("unicorns")
< { ok: 1 }
> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e583e31f2bccbdcacf3466')
    }
  }
> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584991f2bccbdcacf3467')
    }
  }
> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e5849f1f2bccbdcacf3468')
    }
  }
```

```
> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584a51f2bccbdcacf3469')
    }
  }
> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584ae1f2bccbdcacf346a')
    }
  }
> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584b51f2bccbdcacf346b')
    }
  }
```

```
> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584ba1f2bccbdcacf346c')
    }
  }
> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584bf1f2bccbdcacf346d')
    }
  }
> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584c51f2bccbdcacf346e')
    }
  }
```

```
> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584cc1f2bccbdcacf346f')
    }
  }
> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584d01f2bccbdcacf3470')
    }
  }
> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e584d51f2bccbdcacf3471')
    }
  }
```

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

```
> document={name: 'Dunx', loves: ['grape', 'watermelon'], weight: 740, gender: 'm', vampires: 165}
< {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 740,
    gender: 'm',
    vampires: 165
  }
> db.unicorns.insert(document)
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e586c91f2bccbdcacf3472')
    }
  }
```

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

```
> db.unicorns.find()
< {
    _id: ObjectId('65e583e31f2bccbdcacf3466'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('65e584991f2bccbdcacf3467'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
```

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})
< {
    _id: ObjectId('65e586c91f2bccbdcacf3472'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 740,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('65e583e31f2bccbdcacf3466'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
```

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
< {
    _id: ObjectId('65e584991f2bccbdcacf3467'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('65e584b51f2bccbdcacf346b'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
    _id: ObjectId('65e584c51f2bccbdcacf346e'),
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
```

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

```
> db.unicorns.findOne({gender: 'f', loves:'carrot'})
< {
    _id: ObjectId('65e584991f2bccbdcacf3467'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

```
> db.unicorns.find({gender: 'f', loves:'carrot'}).limit(1)
< {
    _id: ObjectId('65e584991f2bccbdcacf3467'),
    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 предпотениях и поле.

```
> db.unicorns.find({gender: 'm'},{gender: 0, loves: 0}).sort({name:1})
< {
    _id: ObjectId('65e586c91f2bccbdcacf3472'),
    name: 'Dunx',
    weight: 740,
    vampires: 165
  }
  {
    _id: ObjectId('65e583e31f2bccbdcacf3466'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
  {
    _id: ObjectId('65e584ba1f2bccbdcacf346c'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
  {
    _id: ObjectId('65e584d01f2bccbdcacf3470'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  }
  {
    _id: ObjectId('65e584bf1f2bccbdcacf346d'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  }
  {
    _id: ObjectId('65e584a51f2bccbdcacf3469'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find().sort({$natural: -1})
< {
    _id: ObjectId('65e586c91f2bccbdcacf3472'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 740,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('65e584d51f2bccbdcacf3471'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
  {
```

Практическое задание 2.1.4:

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

Практическое задание 2.3.1:

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

```
> db.unicorns.find({weight: {$lt:700, $gt: 500}, gender: 'f'},{_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: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
  {
    name: 'Nimue',
    loves: [
      'grape',
```

Практическое задание 2.3.2:

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

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

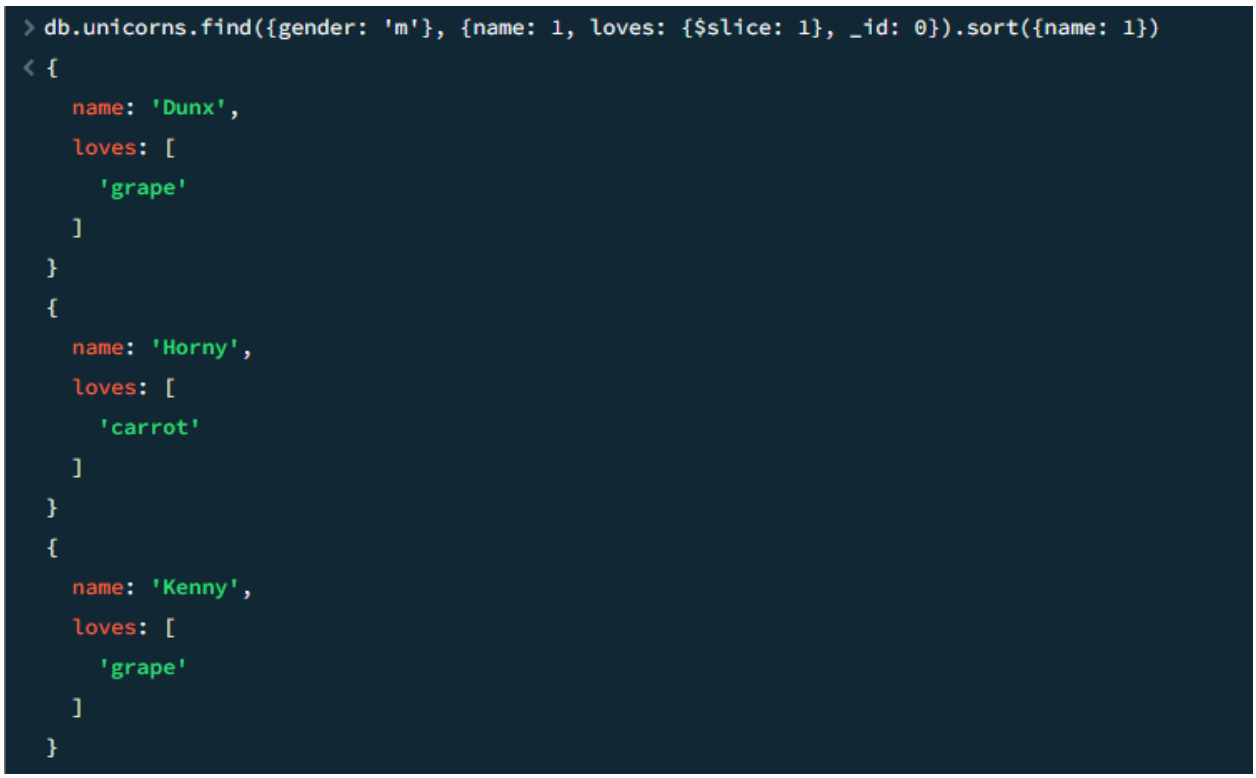
Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ `vampires`.

```
> db.unicorns.find({vampires: {$exists: false}})
< {
    _id: ObjectId('65e584d51f2bccbdcacf3471'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

Практическое задание 2.3.4:

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



3.1 ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ

Практическое задание 3.1.1:

```
> db.towns.insertMany([
    {
      name: "Punxsutawney",
      population: 6200,
      last_sensus: ISODate("2008-01-31"),
      famous_for: [],
      mayor: {
        name: "Jim Wehrle"
      }
    },
    {
      name: "New York",
      population: 22200000,
      last_sensus: ISODate("2009-07-31"),
      famous_for: ["Statue of Liberty", "food"],
      mayor: {
        name: "Michael Bloomberg",
        party: "I"
      }
    },
    {
```

2. *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"},{mayor:1, name:1, _id:0})
< {
    name: 'New York',
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
```

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

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

Практическое задание 3.1.2:

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

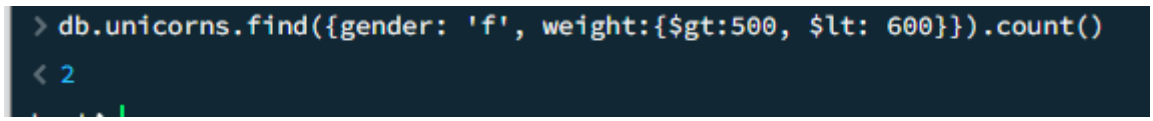
```
> function printMale(){return db.unicorns.find({gender: 'm'})}
< [Function: printMale]
```

2. *Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.*
3. *Вывести результат, используя forEach.*

```
> var cursor = printMale().sort({name: 1}).limit(2);
> cursor.forEach(function(obj){print(obj)});
< {
    _id: ObjectId('65e586c91f2bccbdcacf3472'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 740,
    gender: 'm',
    vampires: 165
  }
< {
    _id: ObjectId('65e583e31f2bccbdcacf3466'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
```

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

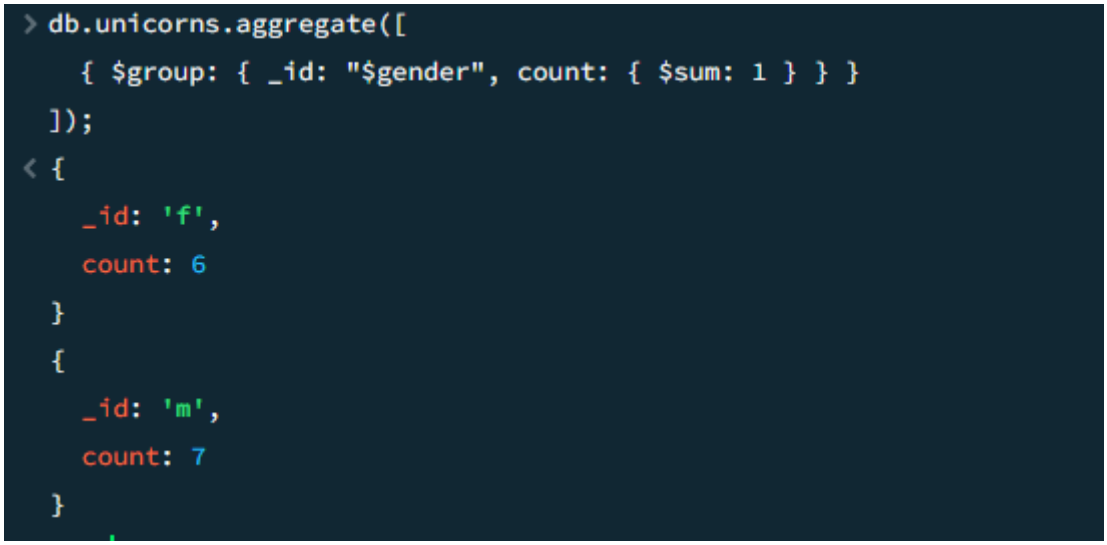


Практическое задание 3.2.2:

Вывести список предпочтений.

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.



Практическое задание 3.3.1:

1. *Выполнить команду:*

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

2. *Проверить содержимое коллекции* `unicorns`.

TypeError: db.unicorns.save is not a function

- Метод save нет в последних версиях mongoBD

```
> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'});
< {
    acknowledged: true,
    insertedId: ObjectId('65e593141f2bccbdcacf3476')
  }
```

```
> db.unicorns.insert({name:'Barny', loves:['grape'], weight: 340, gender: 'm'})
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e5935e1f2bccbdcacf3477')
    }
  }
```

Практическое задание 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`.

```
> db.unicorns.replaceOne({name:"Ayna"}, {name:"Ayna", loves: ["strawberry", "lemon"], weight: 800, gender: "f", vampires: 51})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Практическое задание 3.3.3:

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

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

```
    _id: ObjectId('65e584bf1f2bccbdcacf346d'),
    name: 'Raleigh',
    loves: 'redbull',
    weight: 421,
    gender: 'm',
    vampires: 2
  }
```

Практическое задание 3.3.4:

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

```
> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 9,
    modifiedCount: 9,
    upsertedCount: 0
  }
```

```
    _id: ObjectId('65e583e31f2bccbdcacf3466'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 68
  }
  {
```

Практическое задание 3.3.5:

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

Проверить содержимое коллекции `towns`.

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

```
  _id: ObjectId('65e58cef1f2bccbdcacf3475'),
  name: 'Portland',
  population: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams'
  }
}
```

Практическое задание 3.3.6:

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

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

```
  _id: ObjectId('65e584d01f2bccbdcacf3470'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
```

Практическое задание 3.3.7:

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

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

```
    _id: ObjectId('65e584991f2bccbdcacf3467'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar',
      'lemon'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

Практическое задание 3.4.1:

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"}}
```

2. *Удалите документы с беспартийными мэрами.*
3. *Проверьте содержание коллекции.*
4. *Очистите коллекцию.*
5. *Просмотрите список доступных коллекций.*

```
> db.towns.insertMany([
    {
        name: "Punxsutawney",
        population: 6200,
        last_sensus: ISODate("2008-01-31"),
        famous_for: ["phil the groundhog"],
        mayor: {
            name: "Jim Wehrle"
        }
    },
    {
        name: "New York",
        population: 22200000,
        last_sensus: ISODate("2009-07-31"),
        famous_for: ["status of liberty", "food"],
        mayor: {
            name: "Michael Bloomberg",
            party: "I"
        }
    },
    {
        name: "Portland",
```

```
> db.towns.deleteMany({'mayor.party':{$exists: false}})
< {
    acknowledged: true,
    deletedCount: 3
  }
```

```
> db.towns.find()
< {
    _id: ObjectId('65e58cef1f2bccbdcacf3474'),
    name: 'New York',
    population: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [
      'Statue of Liberty',
      'food'
    ],
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
  {
    _id: ObjectId('65e597d91f2bccbdcacf3479'),
    name: 'New York',
    population: 22200000,
```

```
> db.towns.drop()
< true
```

```
> show collections
< unicorns
```

4.1 ССЫЛКИ В БД

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.places.insertMany([{"_id": "forest","name": "Forest", "description": "coming soon"},{"_id": "dreams", "name": "Dreams", "
< {
    acknowledged: true,
    insertedIds: {
      '0': 'forest',
      '1': 'dreams'
    }
  }
```

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

```
> var forestId = db.places.findOne({_id: "forest"})._id
> var dreamsId = db.places.findOne({_id: "dreams"})._id
```

```
> db.unicorns.updateMany({name: "Ayna"},{$set:{places: {$ref: "places", $id: forestId}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

```
> db.unicorns.updateMany({name: "Aurora"},{$set:{places: {$ref: "places", $id: dreamsId}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

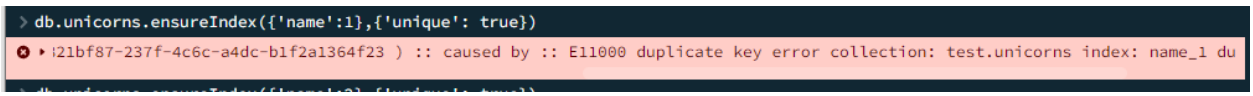
3. *Проверьте содержание коллекции едиорогов.*

```
> db.unicorns.find({name: 'Ayna'})
< {
    _id: ObjectId('65e584b51f2bccbdcacf346b'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 800,
    gender: 'f',
    vampires: 51,
    places: DBRef('places', 'forest')
  }
```

```
> db.unicorns.find({name: 'Aurora'})
< {
    _id: ObjectId('65e584991f2bccbdcacf3467'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar',
      'lemon'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    places: DBRef('places', 'dreams')
  }
```

Практическое задание 4.2.1:

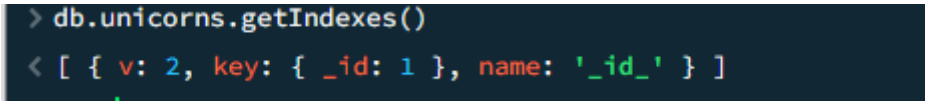
1. Проверьте, можно ли задать для коллекции unicorns индекс для ключа name с флагом unique.

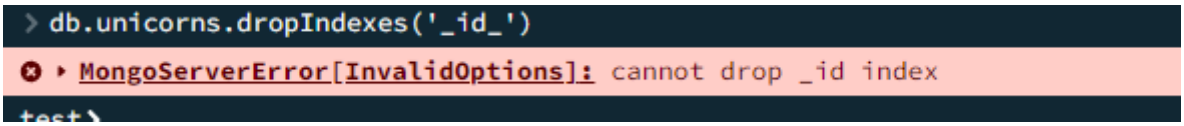


4.3 УПРАВЛЕНИЕ ИНДЕКСАМИ

Практическое задание 4.3.1:

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





4.4 ПЛАН ЗАПРОСА

Практическое задание 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.*
3. *Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра* `executionTimeMillis`*)*
4. *Создайте индекс для ключа* `value`*.*
5. *Получите информацию о всех индексах коллекции* `numbres`*.*
6. *Выполните запрос 2.*
7. *Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?*
8. *Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?*

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

```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65e59e601f2bccbdcad241ba')
    }
  }
> db.numbers.find().count()
< 200000
```

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

```
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 98,
    totalKeysExamined: 0,
    totalDocsExamined: 200000,
    executionStages: {
      stage: 'SORT',
      nReturned: 4,
      executionTimeMillisEstimate: 7,
      works: 200006,
      advanced: 4,
      needTime: 200001,
      needYield: 0,
      saveState: 200,
      restoreState: 200,
      isEOF: 1,
      sortPattern: {
        value: -1
      },
```

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

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 6,
  totalKeysExamined: 4,
  totalDocsExamined: 4,
  executionStages: {
    stage: 'LIMIT',
    nReturned: 4,
    executionTimeMillisEstimate: 0,
    works: 5,
    advanced: 4,
    needTime: 0,
    needYield: 0,
    saveState: 0,
    restoreState: 0,
    isEOF: 1,
    limitAmount: 4,
```

Вывод: После создания индекса время выполнения увеличилось в 16 раз

Вывод: В ходе изучения практических навыков работы с MongoDB, были освоены ключевые аспекты базы данных, такие как CRUD-операции, работа с вложенными объектами, агрегации и изменения данных, а также работа со ссылками и индексами.