Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное

учреждение высшего образования

“Национальный исследовательский университет ИТМО”

Факультет инфокоммуникационных технологий

Лабораторная работа 5.1 Введение в СУБД MongoDB. Установка MongoDB

Лабораторная работа 5.2 Работа с БД в СУБД MongoDB

по дисциплине:

«Проектирование и реализация баз данных»

Выполнил студент:

Вали Насибулла

Группа №K3240

Проверила:

Говорова Марина Михайловна

Санкт-Петербург

2022

ЛАБОРАТОРНАЯ РАБОТА 5.1

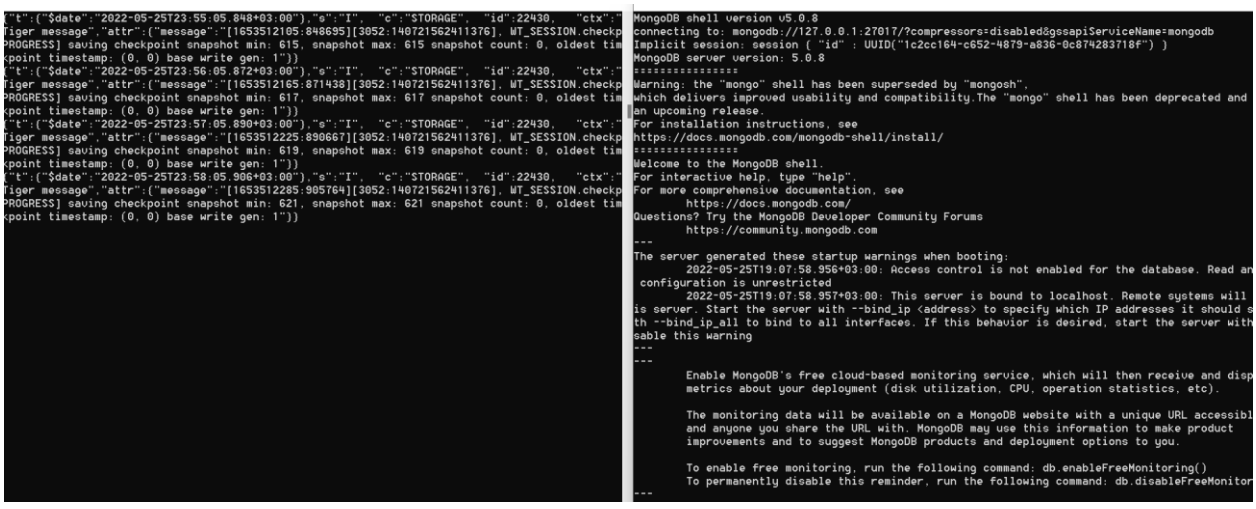
ВВЕДЕНИЕ В СУБД MONGODB. УСТАНОВКА MONGODB. НАЧАЛО РАБОТЫ С БД

Цель: 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:
    a. db.help()
    b. db.help
    c. db.stats()
1. Создайте БД learn.
2. Получите список доступных БД.
3. Создайте коллекцию unicorns, вставив в нее документ `{name: 'Aurora', gender: 'f', weight: 450}`.
4. Просмотрите список текущих коллекций.
5. Переименуйте коллекцию unicorns.
6. Просмотрите статистику коллекции.
7. Удалите коллекцию.
8. Удалите БД learn.

Start working



Db.help()

Showing all methods

```
> db.help()
DB methods:
        db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]
        db.aggregate([pipeline], {options}) - performs a collectionless aggregation on this database; returns a cursor
        db.auth(username, password)
        db.commandHelp(name) returns the help for the command
        db.createUser(userDocument)
        db.createView(name, viewOn, [{$operator: {...}}, ...], {viewOptions})
        db.currentOp() displays currently executing operations in the db
        db.dropDatabase(writeConcern)
        db.dropUser(username)
        db.eval() - deprecated
        db.fsyncLock() flush data to disk and lock server for backups
        db.fsyncUnlock() unlocks server following a db.fsyncLock()
        db.getCollection(cname) same as db['cname'] or db.cname
        db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
        db.getCollectionNames()
        db.getLastError() - just returns the err msg string
        db.getLastErrorObj() - return full status object
        db.getLogComponents()
        db.getMongo() get the server connection object
        db.getMongo().setSecondaryOk() allow queries on a replication secondary server
        db.getName()
        db.getProfilingLevel() - deprecated
        db.getProfilingStatus() - returns if profiling is on and slow threshold
        db.getReplicationInfo()
        db.getSiblingDB(name) get the db at the same server as this one
        db.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server obje
t if set
        db.hostInfo() get details about the server's host
        db.isMaster() check replica primary status
        db.hello() check replica primary status
        db.killOp(opid) kills the current operation in the db
        db.listCommands() lists all the db commands
        db.loadServerScripts() loads all the scripts in db.system.js
        db.logout()
        db.printCollectionStats()
        db.printReplicationInfo()
        db.printShardingStatus()
        db.printSecondaryReplicationInfo()
        db.rotateCertificates(message) - rotates certificates, CRLs, and CA files and logs an optional message
        db.runCommand(cmdObj) run a database command.  if cmdObj is a string, turns it into {cmdObj: 1}
        db.serverStatus()
        db.setLogLevel(level,<component>)
        db.setProfilingLevel(level,slowms) 0=off 1=slow 2=all
        db.setVerboseShell(flag) display extra information in shell output
        db.setWriteConcern(<write concern doc>) - sets the write concern for writes to the db
        db.shutdownServer()
        db.stats()
```

Db.help

```
> db.help
function() {
    print("DB methods:");
    print(
        "\tdb.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]");

    print(
        "\tdb.aggregate([pipeline], {options}) - performs a collectionless aggregation on this database; returns a curso
r");
    print("\tdb.auth(username, password)");
    print("\tdb.commandHelp(name) returns the help for the command");
    print("\tdb.createUser(userDocument)");
    print("\tdb.createView(name, viewOn, [{$operator: {...}}, ...], {viewOptions})");
    print("\tdb.currentOp() displays currently executing operations in the db");
    print("\tdb.dropDatabase(writeConcern)");
    print("\tdb.dropUser(username)");
    print("\tdb.eval() - deprecated");
    print("\tdb.fsyncLock() flush data to disk and lock server for backups");
    print("\tdb.fsyncUnlock() unlocks server following a db.fsyncLock()");
    print("\tdb.getCollection(cname) same as db['cname'] or db.cname");
    print("\tdb.getCollectionInfos([filter]) - returns a list that contains the names and options" +
          " of the db's collections");
    print("\tdb.getCollectionNames()");
    print("\tdb.getLastError() - just returns the err msg string");
    print("\tdb.getLastErrorObj() - return full status object");
    print("\tdb.getLogComponents()");
    print("\tdb.getMongo() get the server connection object");
    print("\tdb.getMongo().setSecondaryOk() allow queries on a replication secondary server");
    print("\tdb.getName()");
    print("\tdb.getProfilingLevel() - deprecated");
    print("\tdb.getProfilingStatus() - returns if profiling is on and slow threshold");
    print("\tdb.getReplicationInfo()");
    print("\tdb.getSiblingDB(name) get the db at the same server as this one");
    print(
        "\tdb.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server ob
ject if set");
    print("\tdb.hostInfo() get details about the server's host");
    print("\tdb.isMaster() check replica primary status");
    print("\tdb.hello() check replica primary status");
    print("\tdb.killOp(opid) kills the current operation in the db");
    print("\tdb.listCommands() lists all the db commands");
    print("\tdb.loadServerScripts() loads all the scripts in db.system.js");
    print("\tdb.logout()");
    print("\tdb.printCollectionStats()");
    print("\tdb.printReplicationInfo()");
    print("\tdb.printShardingStatus()");
    print("\tdb.printSecondaryReplicationInfo()");
    print(
        "\tdb.rotateCertificates(message) - rotates certificates, CRLs, and CA files and logs an optional message");
```

Db.stats()

```
> db.stats()
{
        "db" : "learn",
        "collections" : 0,
        "views" : 0,
        "objects" : 0,
        "avgObjSize" : 0,
        "dataSize" : 0,
        "storageSize" : 0,
        "totalSize" : 0,
        "indexes" : 0,
        "indexSize" : 0,
        "scaleFactor" : 1,
        "fileSize" : 0,
        "fsUsedSize" : 0,
        "fsTotalSize" : 0,
        "ok" : 1
}
>
```

Use learn for creating database

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

Database by the name of learn was not shown in this list because it there was nothing in it.

In order to display learn database, we need to create a collection with documents in the same database as shown below

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

After adding collection to database learn. And using the show dbs command now visible.

```
> show dbs
admin   0.000GB
config  0.000GB
learn   0.000GB
local   0.000GB
>
```

In below image, I am switched to “learn” database, secondly created collection by the name of “unicorns”, and then insert data according to task, and then list the all existing collections.

```
> use learn
switched to db learn
> db.createCollection("unicorns")
{ "ok" : 1 }
> db.unicorns.insert({name: 'Aurora', gender: 'f', weight: 450});
WriteResult({ "nInserted" : 1 })
> show collections
unicorns
>
```

Using renameCollection method I renamed the collection to renamedunicorns

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

Stats of database

For checking the stats of collection we can use method db.collection.stats() but it will show so long information.

```
> db.stats()
{
        "db" : "learn",
        "collections" : 1,
        "views" : 0,
        "objects" : 1,
        "avgObjSize" : 69,
        "dataSize" : 69,
        "storageSize" : 20480,
        "indexes" : 1,
        "indexSize" : 20480,
        "totalSize" : 40960,
        "scaleFactor" : 1,
        "fsUsedSize" : 119967522816,
        "fsTotalSize" : 128849014784,
        "ok" : 1
}
>
```

For dropping collection

```
> db.renamedunicorns.drop()
true
> show collections
>
```

For dropping database

```
> use learn
switched to db learn
> db.dropDatabase()
{ "ok" : 1 }
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

ЛАБОРАТОРНАЯ РАБОТА 5.2

Работа с БД в СУБД MongoDB

Цель: 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.

ПРАКТИЧЕСКАЯ ЧАСТЬ

8.1 CRUD-ОПЕРАЦИИ В СУБД MONGODB. ВСТАВКА ДАННЫХ. ВЫБОРКА ДАННЫХ

1. ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ

Практическое задание 8.1.1:

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'});
```

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

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

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

```
> use learn
switched to db learn
> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
WriteResult({ "nInserted" : 1 })
> document = ({name: 'Dunx', loves: ['grape','watermelon'], weight:704, gender: 'm', vampires: 165})
{
        "name" : "Dunx",
        "loves" : [
                "grape",
                "watermelon"
        ],
        "weight" : 704,
        "gender" : "m",
        "vampires" : 165
}
> db.unicorns.insert(document)
WriteResult({ "nInserted" : 1 })
```

Find()

```
> db.unicorns.find()
{ "_id" : ObjectId("628f2c8dbaff44080e5c9956"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gen
der" : "m", "vampires" : 63 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9957"), "name" : "Aurora", "loves" : [ "carrot", "grape" ], "weight" : 450, "gen
der" : "f", "vampires" : 43 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9958"), "name" : "Unicrom", "loves" : [ "energon", "redbull" ], "weight" : 984,
"gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9959"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender"
: "m", "vampires" : 99 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995a"), "name" : "Solnara", "loves" : [ "apple", "carrot", "chocolate" ], "weigh
t" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995b"), "name" : "Ayna", "loves" : [ "strawberry", "lemon" ], "weight" : 733, "g
ender" : "f", "vampires" : 40 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995c"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gende
r" : "m", "vampires" : 39 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995d"), "name" : "Raleigh", "loves" : [ "apple", "sugar" ], "weight" : 421, "gen
der" : "m", "vampires" : 2 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995e"), "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 601, "g
ender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995f"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "
gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9960"), "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gend
er" : "f" }
{ "_id" : ObjectId("628f2deebaff44080e5c9961"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "g
ender" : "m", "vampires" : 165 }
>
```

8.2.2 ВЫБОРКА ДАННЫХ ИЗ БД

Как было показано выше, наиболее простой способ получения содержимого БД представляет использование метода find. В большинстве запросов возникает необходимость извлечения только тех документов, которые удовлетворяют заданным критериям.

Практическое задание 8.1.2:

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

All males gender=’m’;

```
> db.unicorns.find({gender: "m"})
{ "_id" : ObjectId("628f2c8dbaff44080e5c9956"), "name" : "Horny", "loves" : [ "carrot", "papay
der" : "m", "vampires" : 63 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9958"), "name" : "Unicrom", "loves" : [ "energon", "re
"gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9959"), "name" : "Roooooodles", "loves" : [ "apple" ],
: "m", "vampires" : 99 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995c"), "name" : "Kenny", "loves" : [ "grape", "lemon"
r" : "m", "vampires" : 39 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995d"), "name" : "Raleigh", "loves" : [ "apple", "suga
der" : "m", "vampires" : 2 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995f"), "name" : "Pilot", "loves" : [ "apple", "waterm
gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("628f2deebaff44080e5c9961"), "name" : "Dunx", "loves" : [ "grape", "waterme
ender" : "m", "vampires" : 165 }
>
```

All females to the first three individuals and sort by name.

```
> db.unicorns.find({gender: "f"}).sort({name:1}).limit(3)
{ "_id" : ObjectId("628f2c8dbaff44080e5c9957"), "name" : "Aurora", "loves" : [ "carrot", "grap
der" : "f", "vampires" : 43 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995b"), "name" : "Ayna", "loves" : [ "strawberry", "le
ender" : "f", "vampires" : 40 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995e"), "name" : "Leia", "loves" : [ "apple", "waterme
ender" : "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("628f2c8dbaff44080e5c9957"),
        "name" : "Aurora",
        "loves" : [
                "carrot",
                "grape"
        ],
        "weight" : 450,
        "gender" : "f",
        "vampires" : 43
}
>
```

Практическое задание 8.1.3:

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

```
> db.unicorns.find({gender: "m"}, {"loves":0, "gender": 0})
{ "_id" : ObjectId("628f2c8dbaff44080e5c9956"), "name" : "Horny", "weight" : 600, "vampires" :
{ "_id" : ObjectId("628f2c8dbaff44080e5c9958"), "name" : "Unicrom", "weight" : 984, "vampires"
{ "_id" : ObjectId("628f2c8dbaff44080e5c9959"), "name" : "Roooooodles", "weight" : 575, "vampi
{ "_id" : ObjectId("628f2c8dbaff44080e5c995c"), "name" : "Kenny", "weight" : 690, "vampires" :
{ "_id" : ObjectId("628f2c8dbaff44080e5c995d"), "name" : "Raleigh", "weight" : 421, "vampires"
{ "_id" : ObjectId("628f2c8dbaff44080e5c995f"), "name" : "Pilot", "weight" : 650, "vampires" :
{ "_id" : ObjectId("628f2deebaff44080e5c9961"), "name" : "Dunx", "weight" : 704, "vampires" :
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find().sort({$natural: -1})
{ "_id" : ObjectId("628f2deebaff44080e5c9961"), "name" : "Dunx", "loves" : [ "grape", "waterme
ender" : "m", "vampires" : 165 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9960"), "name" : "Nimue", "loves" : [ "grape", "carrot
er" : "f" }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995f"), "name" : "Pilot", "loves" : [ "apple", "waterm
gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995e"), "name" : "Leia", "loves" : [ "apple", "waterme
ender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995d"), "name" : "Raleigh", "loves" : [ "apple", "suga
der" : "m", "vampires" : 2 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995c"), "name" : "Kenny", "loves" : [ "grape", "lemon"
r" : "m", "vampires" : 39 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995b"), "name" : "Ayna", "loves" : [ "strawberry", "le
ender" : "f", "vampires" : 40 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c995a"), "name" : "Solnara", "loves" : [ "apple", "carr
t" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9959"), "name" : "Roooooodles", "loves" : [ "apple" ],
: "m", "vampires" : 99 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9958"), "name" : "Unicrom", "loves" : [ "energon", "re
"gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9957"), "name" : "Aurora", "loves" : [ "carrot", "grap
der" : "f", "vampires" : 43 }
{ "_id" : ObjectId("628f2c8dbaff44080e5c9956"), "name" : "Horny", "loves" : [ "carrot", "papay
der" : "m", "vampires" : 63 }
>
```

Практическое задание 8.1.5:

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

```
> db.unicorns.find({},{_id:0, loves: {$slice :1}})
{ "name" : "Horny", "loves" : [ "carrot" ], "weight" : 600, "gender" : "m", "vampires" : 63 }
{ "name" : "Aurora", "loves" : [ "carrot" ], "weight" : 450, "gender" : "f", "vampires" : 43 }
{ "name" : "Unicrom", "loves" : [ "energon" ], "weight" : 984, "gender" : "m", "vampires" : 18
{ "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" :
{ "name" : "Solnara", "loves" : [ "apple" ], "weight" : 550, "gender" : "f", "vampires" : 80 }
{ "name" : "Ayna", "loves" : [ "strawberry" ], "weight" : 733, "gender" : "f", "vampires" : 40
{ "name" : "Kenny", "loves" : [ "grape" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "name" : "Raleigh", "loves" : [ "apple" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "name" : "Leia", "loves" : [ "apple" ], "weight" : 601, "gender" : "f", "vampires" : 33 }
{ "name" : "Pilot", "loves" : [ "apple" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "name" : "Nimue", "loves" : [ "grape" ], "weight" : 540, "gender" : "f" }
{ "name" : "Dunx", "loves" : [ "grape" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
>
```

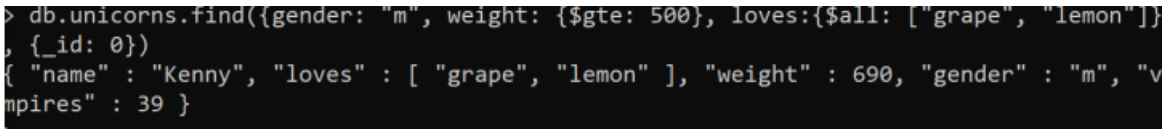
8.2.3 ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Практическое задание 8.1.6:

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

Практическое задание 8.1.7:

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



Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ `vampires`.

Практическое задание 8.1.9:

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

```
db.unicorns.find({},{loves : {$slice: 1}}).sort({name:1})
"_id" : ObjectId("628f3c351a95973f6d218082"), "name" : "Aurora", "loves" : [ "carrot" ], "weig
 "vampires" : 43 }
"_id" : ObjectId("628f3c351a95973f6d218086"), "name" : "Ayna", "loves" : [ "strawberry" ], "we
", "vampires" : 40 }
"_id" : ObjectId("628f3c351a95973f6d218081"), "name" : "Horny", "loves" : [ "carrot" ], "weigh
"vampires" : 63 }
"_id" : ObjectId("628f3c351a95973f6d218087"), "name" : "Kenny", "loves" : [ "grape" ], "weight
vampires" : 39 }
"_id" : ObjectId("628f3c351a95973f6d218089"), "name" : "Leia", "loves" : [ "apple" ], "weight"
ampires" : 33 }
"_id" : ObjectId("628f3c351a95973f6d21808b"), "name" : "Nimue", "loves" : [ "grape" ], "weight

"_id" : ObjectId("628f3c351a95973f6d21808a"), "name" : "Pilot", "loves" : [ "apple" ], "weight
vampires" : 54 }
"_id" : ObjectId("628f3c351a95973f6d218088"), "name" : "Raleigh", "loves" : [ "apple" ], "weig
 "vampires" : 2 }
"_id" : ObjectId("628f3c351a95973f6d218084"), "name" : "Roooooodles", "loves" : [ "apple" ], "
"m", "vampires" : 99 }
"_id" : ObjectId("628f3c351a95973f6d218085"), "name" : "Solnara", "loves" : [ "apple" ], "weig
 "vampires" : 80 }
"_id" : ObjectId("628f3c351a95973f6d218083"), "name" : "Unicrom", "loves" : [ "energon" ], "we
", "vampires" : 182 }
```

8.2 ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB. ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ

Практическое задание 8.2.1:

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

```
> db.createCollection("towns")
{ "ok" : 1 }
> db.towns.insert({name: "Punxsutawney ",
... populatiuon: 6200,
... last_sensus: ISODate("2008-01-31"),
... famous_for: [""],
... mayor: {
...     name: "Jim Wehrle"
... }}
... )
WriteResult({ "nInserted" : 1 })
> 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"}}
... )
WriteResult({ "nInserted" : 1 })
> db.towns.insert({name: "Portland",
... populatiuon: 528000,
... last_sensus: ISODate("2009-07-20"),
... famous_for: ["beer", "food"],
... mayor: {
...     name: "Sam Adams",
... party: "D"}}
... )
WriteResult({ "nInserted" : 1 })
>
```

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"}, {_id:0, populatiuon : 0, last_sensus: 0, famous_for :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" : null}, {_id:0, populatiuon : 0, last_sensus: 0, famous_for :0
{ "name" : "Punxsutawney ", "mayor" : { "name" : "Jim Wehrle" } }
>
```

1. ИСПОЛЬЗОВАНИЕ JAVASCRIPT

Практическое задание 8.2.2:

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

```
> myfunction = function() { return this.gender == "m"}
function() { return this.gender == "m"}
> db.unicorns.find(myfunction)
{ "_id" : ObjectId("628f3c351a95973f6d218081"), "name" : "Horny", "loves" : [ "carrot", "papay
der" : "m", "vampires" : 63 }
{ "_id" : ObjectId("628f3c351a95973f6d218083"), "name" : "Unicrom", "loves" : [ "energon", "re
"gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("628f3c351a95973f6d218084"), "name" : "Roooooodles", "loves" : [ "apple" ],
: "m", "vampires" : 99 }
{ "_id" : ObjectId("628f3c351a95973f6d218087"), "name" : "Kenny", "loves" : [ "grape", "lemon"
r" : "m", "vampires" : 39 }
{ "_id" : ObjectId("628f3c351a95973f6d218088"), "name" : "Raleigh", "loves" : [ "apple", "suga
der" : "m", "vampires" : 2 }
{ "_id" : ObjectId("628f3c351a95973f6d21808a"), "name" : "Pilot", "loves" : [ "apple", "waterm
gender" : "m", "vampires" : 54 }
>
```

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

```
> var mycursor= db.unicorns.find();null
null
> mycursor.limit(2);null
null
> mycursor.sort({name: 1});null
null
>
```

3. *Вывести результат, используя forEach.*

```
> mycursor.forEach(function(obj) { print(obj.name);})
Ayna
>
```

Содержание коллекции единорогов `unicorns`*:*

8.2.4 АГРЕГИРОВАННЫЕ ЗАПРОСЫ

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 8.2.4:

Вывести список предпочтений.

```
> db.unicorns.distinct("name")
[
        "Aurora",
        "Ayna",
        "Horny",
        "Kenny",
        "Leia",
        "Nimue",
        "Pilot",
        "Raleigh",
        "Roooooodles",
        "Solnara",
        "Unicrom"
]
>
```

Практическое задание 8.2.5:

Посчитать количество особей единорогов обоих полов.

```
> db.unicorns.aggregate({"$group":{_id:"$gender", count:{$sum:1}}})
{ "_id" : "m", "count" : 6 }
{ "_id" : "f", "count" : 5 }
>
```

8.2.5 РЕДАКТИРОВАНИЕ ДАННЫХ

Практическое задание 8.2.6:

1. *Выполнить команду:*

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

2. *Проверить содержимое коллекции* `unicorns`.

```
> db.unicorns.save({name: 'Barny',loves: ['grape'],weight: 340, gender: 'm'})
WriteResult({ "nInserted" : 1 })
> db.unicorns.find()
{ "_id" : ObjectId("628f3c351a95973f6d218081"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gen
der" : "m", "vampires" : 63 }
{ "_id" : ObjectId("628f3c351a95973f6d218082"), "name" : "Aurora", "loves" : [ "carrot", "grape" ], "weight" : 450, "gen
der" : "f", "vampires" : 43 }
{ "_id" : ObjectId("628f3c351a95973f6d218083"), "name" : "Unicrom", "loves" : [ "energon", "redbull" ], "weight" : 984,
"gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("628f3c351a95973f6d218084"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender"
: "m", "vampires" : 99 }
{ "_id" : ObjectId("628f3c351a95973f6d218085"), "name" : "Solnara", "loves" : [ "apple", "carrot", "chocolate" ], "weigh
t" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("628f3c351a95973f6d218086"), "name" : "Ayna", "loves" : [ "strawberry", "lemon" ], "weight" : 733, "g
ender" : "f", "vampires" : 40 }
{ "_id" : ObjectId("628f3c351a95973f6d218087"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gende
r" : "m", "vampires" : 39 }
{ "_id" : ObjectId("628f3c351a95973f6d218088"), "name" : "Raleigh", "loves" : [ "apple", "sugar" ], "weight" : 421, "gen
der" : "m", "vampires" : 2 }
{ "_id" : ObjectId("628f3c351a95973f6d218089"), "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 601, "g
ender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("628f3c351a95973f6d21808a"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "
gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("628f3c351a95973f6d21808b"), "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gend
er" : "f" }
{ "_id" : ObjectId("628f44151a95973f6d21808f"), "name" : "Barny", "loves" : [ "grape" ], "weight" : 340, "gender" : "m"
}
>
```

Практическое задание 8.2.7:

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

```
> db.unicorns.update({name: "Ayna"}, {name: "Ayna", weight: 800, vampires: 51})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Практическое задание 8.2.8:

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

```
> db.unicorns.update({name: "Raleigh"}, {$set: {loves: "RedBull"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Практическое задание 8.2.9:

1. *Всем самцам единорогов увеличить количество убитых вапмиров на 5.*

```
> db.unicorns.update({gender: "m"}, {$inc: {vampires: 5}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.find()
{ "_id" : ObjectId("628f3c351a95973f6d218081"), "name" : "Horny", "loves" : [ "carrot", "papay
der" : "m", "vampires" : 68 }
{ "_id" : ObjectId("628f3c351a95973f6d218082"), "name" : "Aurora", "loves" : [ "carrot", "grap
der" : "f", "vampires" : 43 }
{ "_id" : ObjectId("628f3c351a95973f6d218083"), "name" : "Unicrom", "loves" : [ "energon", "re
"gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("628f3c351a95973f6d218084"), "name" : "Roooooodles", "loves" : [ "apple" ],
: "m", "vampires" : 99 }
{ "_id" : ObjectId("628f3c351a95973f6d218085"), "name" : "Solnara", "loves" : [ "apple", "carr
t" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("628f3c351a95973f6d218086"), "name" : "Ayna", "weight" : 800, "vampires" :
{ "_id" : ObjectId("628f3c351a95973f6d218087"), "name" : "Kenny", "loves" : [ "grape", "lemon"
r" : "m", "vampires" : 39 }
{ "_id" : ObjectId("628f3c351a95973f6d218088"), "name" : "Raleigh", "loves" : "RedBull", "weig
 "vampires" : 2 }
{ "_id" : ObjectId("628f3c351a95973f6d218089"), "name" : "Leia", "loves" : [ "apple", "waterme
ender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("628f3c351a95973f6d21808a"), "name" : "Pilot", "loves" : [ "apple", "waterm
gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("628f3c351a95973f6d21808b"), "name" : "Nimue", "loves" : [ "grape", "carrot
er" : "f" }
{ "_id" : ObjectId("628f44151a95973f6d21808f"), "name" : "Barny", "loves" : [ "grape" ], "weig
}
>
```

Практическое задание 8.2.10:

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

```
> db.towns.update({name: "Portland"}, {$unset: {mayor: 1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.towns.find()
{ "_id" : ObjectId("628f3d961a95973f6d21808c"), "name" : "Punxsutawney ", "populatiuon" : 6200
"2008-01-31T00:00:00Z"), "famous_for" : [ "" ], "mayor" : { "name" : "Jim Wehrle" } }
{ "_id" : ObjectId("628f3dec1a95973f6d21808d"), "name" : "New York", "populatiuon" : 22200000,
2009-07-31T00:00:00Z"), "famous_for" : [ "status of liberty", "food" ], "mayor" : { "name" : "
" : "I" } }
{ "_id" : ObjectId("628f3e071a95973f6d21808e"), "name" : "Portland", "populatiuon" : 528000, "
09-07-20T00:00:00Z"), "famous_for" : [ "beer", "food" ] }
>
```

Практическое задание 8.2.11:

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

```
> db.unicorns.update({name: "Pilot"}, {$push: {loves: "chocolate"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Практическое задание 8.2.12:

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

```
> db.unicorns.update({name: "Aurora"}, {$addToSet: {loves: ["sugar,lemon"]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

8.2.6 УДАЛЕНИЕ ДАННЫХ ИЗ КОЛЛЕКЦИИ

Практическое задание 8.2.13:

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

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

```
> db.towns.remove({"mayor.party":null},true)
WriteResult({ "nRemoved" : 1 })
> db.towns.find()
{ "_id" : ObjectId("628f3dec1a95973f6d21808d"), "name" : "New York", "populatiuon" : 22200000,
2009-07-31T00:00:00Z"), "famous_for" : [ "status of liberty", "food" ], "mayor" : { "name" : "
" : "I" } }
{ "_id" : ObjectId("628f3e071a95973f6d21808e"), "name" : "Portland", "populatiuon" : 528000, "
09-07-20T00:00:00Z"), "famous_for" : [ "beer", "food" ] }
>
```

```
> show collections
towns
unicorns
> db.towns.remove({})
WriteResult({ "nRemoved" : 2 })
> db.stats()
{
        "db" : "learn",
        "collections" : 2,
        "views" : 0,
        "objects" : 12,
        "avgObjSize" : 120.5,
        "dataSize" : 1446,
        "storageSize" : 73728,
        "indexes" : 2,
        "indexSize" : 61440,
        "totalSize" : 135168,
        "scaleFactor" : 1,
        "fsUsedSize" : 119965495296,
        "fsTotalSize" : 128849014784,
        "ok" : 1
}
>
```

8.3 ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

8.3.1 ССЫЛКИ В БД

Практическое задание 8.3.1:

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

```
> db.home.insert({_id: "su", name: "sun"})
WriteResult({ "nInserted" : 1 })
> db.home.insert({_id: "mo", name: "moon"})
WriteResult({ "nInserted" : 1 })
> db.home.insert({_id: "pl", name: "plannet"})
WriteResult({ "nInserted" : 1 })
> db.home.find()
{ "_id" : "su", "name" : "sun" }
{ "_id" : "mo", "name" : "moon" }
{ "_id" : "pl", "name" : "plannet" }
>
```

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

```
> db.unicorns.update({_id:objectId("628f3c351a95973f6d218081")}, {$set:{home:{$ref:"home",id:"
uncaught exception: ReferenceError: objectId is not defined :
@(shell):1:21
> db.unicorns.update({_id:ObjectId("628f3c351a95973f6d218081")}, {$set:{home:{$ref:"home",id:"
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> var Horny = db.unicorns.findOne({"_id": ObjectId("628f3c351a95973f6d218081")})
> Horny
{
        "_id" : ObjectId("628f3c351a95973f6d218081"),
        "name" : "Horny",
        "loves" : [
                "carrot",
                "papaya"
        ],
        "weight" : 600,
        "gender" : "m",
        "vampires" : 68,
        "home" : {
                "$ref" : "home",
                "id" : "su"
        }
}
>
```

3. Output

```
> db.unicorns.find()
{ "_id" : ObjectId("628f3c351a95973f6d218081"), "name" : "Horny", "loves" : [ "carrot", "papay
der" : "m", "vampires" : 68, "home" : { "$ref" : "home", "id" : "su" } }
{ "_id" : ObjectId("628f3c351a95973f6d218082"), "name" : "Aurora", "loves" : [ "carrot", "grap
weight" : 450, "gender" : "f", "vampires" : 43 }
{ "_id" : ObjectId("628f3c351a95973f6d218083"), "name" : "Unicrom", "loves" : [ "energon", "re
"gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("628f3c351a95973f6d218084"), "name" : "Roooooodles", "loves" : [ "apple" ],
: "m", "vampires" : 99 }
{ "_id" : ObjectId("628f3c351a95973f6d218085"), "name" : "Solnara", "loves" : [ "apple", "carr
t" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("628f3c351a95973f6d218086"), "name" : "Ayna", "weight" : 800, "vampires" :
{ "_id" : ObjectId("628f3c351a95973f6d218087"), "name" : "Kenny", "loves" : [ "grape", "lemon"
r" : "m", "vampires" : 39 }
{ "_id" : ObjectId("628f3c351a95973f6d218088"), "name" : "Raleigh", "loves" : "RedBull", "weig
 "vampires" : 2 }
{ "_id" : ObjectId("628f3c351a95973f6d218089"), "name" : "Leia", "loves" : [ "apple", "waterme
ender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("628f3c351a95973f6d21808a"), "name" : "Pilot", "loves" : [ "apple", "waterm
ght" : 650, "gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("628f3c351a95973f6d21808b"), "name" : "Nimue", "loves" : [ "grape", "carrot
er" : "f" }
{ "_id" : ObjectId("628f44151a95973f6d21808f"), "name" : "Barny", "loves" : [ "grape" ], "weig
}
>
```

8.3.2 НАСТРОЙКА ИНДЕКСОВ

Практическое задание 8.3.2:

1. *Проверьте, можно ли задать для коллекции* `unicorns` *индекс для ключа* `name` *с флагом* `unique`.
2. *Содержание коллекции единорогов* `unicorns`*:*

```
> db.unicorns.ensureIndex({"name" : 1}, {"unique" : true});
uncaught exception: TypeError: db.unicorns.ensureIndex is not a function :
@(shell):1:1
>
```

8.3.3 УПРАВЛЕНИЕ ИНДЕКСАМИ

Практическое задание 8.3.3:

1. *Получите информацию о всех индексах коллекции* `unicorns`.

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

8.3.4 ПЛАН ЗАПРОСА

Практическое задание 8.3.4:

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

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

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

```
> var cursor = db.numbers.find(); null
null
> for (i=0; i< 100000; i++){db.numbers.insert({value: i})}
WriteResult({ "nInserted" : 1 })
```

```
> db.numbers.find()
{ "_id" : ObjectId("628f515ab16569b0ba565fe7"), "value" : 0 }
{ "_id" : ObjectId("628f515ab16569b0ba565fe8"), "value" : 1 }
{ "_id" : ObjectId("628f515ab16569b0ba565fe9"), "value" : 2 }
{ "_id" : ObjectId("628f515ab16569b0ba565fea"), "value" : 3 }
{ "_id" : ObjectId("628f515ab16569b0ba565feb"), "value" : 4 }
{ "_id" : ObjectId("628f515ab16569b0ba565fec"), "value" : 5 }
{ "_id" : ObjectId("628f515ab16569b0ba565fed"), "value" : 6 }
{ "_id" : ObjectId("628f515ab16569b0ba565fee"), "value" : 7 }
{ "_id" : ObjectId("628f515ab16569b0ba565fef"), "value" : 8 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff0"), "value" : 9 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff1"), "value" : 10 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff2"), "value" : 11 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff3"), "value" : 12 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff4"), "value" : 13 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff5"), "value" : 14 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff6"), "value" : 15 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff7"), "value" : 16 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff8"), "value" : 17 }
{ "_id" : ObjectId("628f515ab16569b0ba565ff9"), "value" : 18 }
{ "_id" : ObjectId("628f515ab16569b0ba565ffa"), "value" : 19 }
Type "it" for more
>
```

2 Выберите последних четыре документа.

```
> db.numbers.find().sort({value: -1}).limit(4);
{ "_id" : ObjectId("628f5175b16569b0ba57e686"), "value" : 99999 }
{ "_id" : ObjectId("628f5175b16569b0ba57e685"), "value" : 99998 }
{ "_id" : ObjectId("628f5175b16569b0ba57e684"), "value" : 99997 }
{ "_id" : ObjectId("628f5175b16569b0ba57e683"), "value" : 99996 }
```

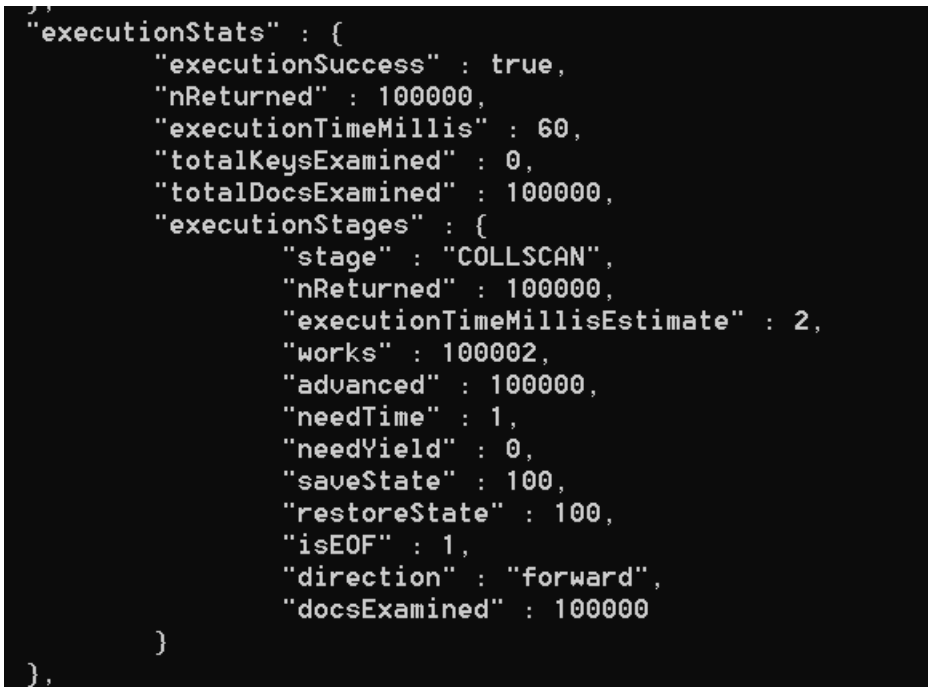
3.time of execution

```
> db.numbers.find().sort({value: -1}).limit(4).explain();
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "test.numbers",
                "indexFilterSet" : false,
                "parsedQuery" : {

                },
                "queryHash" : "E22F2B9F",
                "planCacheKey" : "378A1826",
                "maxIndexedOrSolutionsReached" : false,
                "maxIndexedAndSolutionsReached" : false,
                "maxScansToExplodeReached" : false,
                "winningPlan" : {
                        "stage" : "SORT",
                        "sortPattern" : {
                                "value" : -1
                        },
                        "memLimit" : 104857600,
                        "limitAmount" : 4,
                        "type" : "simple",
                        "inputStage" : {
                                "stage" : "COLLSCAN",
                                "direction" : "forward"
                        }
                },
                "rejectedPlans" : [ ]
        },
        "command" : {
                "find" : "numbers",
                "filter" : {

                },
                "limit" : 4,
                "singleBatch" : false,
                "sort" : {
                        "value" : -1
                },
                "$db" : "test"
        },
        "serverInfo" : {
                "host" : "LAPTOP-8HD61RBF",
```

```
    "executionStats" : {
            "executionSuccess" : true,
            "nReturned" : 4,
            "executionTimeMillis" : 193,
            "totalKeysExamined" : 0,
            "totalDocsExamined" : 100000,
            "executionStages" : {
                    "stage" : "SORT",
                    "nReturned" : 4,
                    "executionTimeMillisEstimate" : 8,
                    "works" : 100007,
                    "advanced" : 4,
                    "needTime" : 100002,
                    "needYield" : 0,
                    "saveState" : 100,
                    "restoreState" : 100,
                    "isEOF" : 1,
                    "sortPattern" : {
                            "value" : -1
                    },
                    "memLimit" : 104857600,
                    "limitAmount" : 4,
                    "type" : "simple",
                    "totalDataSizeSorted" : 6900000,
                    "usedDisk" : false,
                    "inputStage" : {
                            "stage" : "COLLSCAN",
                            "nReturned" : 100000,
                            "executionTimeMillisEstimate" : 4,
                            "works" : 100002,
                            "advanced" : 100000,
                            "needTime" : 1,
                            "needYield" : 0,
                            "saveState" : 100,
                            "restoreState" : 100,
                            "isEOF" : 1,
                            "direction" : "forward",
                            "docsExamined" : 100000
                    }
            }
    },
```



```
> db.numbers.createIndex({"value" :1})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
>
```


"executionStats" : {
"executionSuccess" : true,
"nReturned" : 100000,
"executionTimeMillis" : 60,
"totalKeysExamined" : 0,
"totalDocsExamined" : 100000,
"executionStages" : {
"stage" : "COLLSCAN",
"nReturned" : 100000,
"executionTimeMillisEstimate" : 2,
"works" : 100002,
"advanced" : 100000,
"needTime" : 1,
"needYield" : 0,
"saveState" : 100,
"restoreState" : 100,
"isEOF" : 1,
"direction" : "forward",
"docsExamined" : 100000
}
},