

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ № 5

по теме:

«Работа с БД в СУБД MongoDB»

по дисциплине: Проектирование и реализация баз данных

Специальность:

45.03.04 Интеллектуальные системы в гуманитарной сфере

Проверила:

Говорова М.М.

Дата: «...» ... 2023 г.

Оценка _____

Выполнил:

студент группы К32422

Малаев С.Г.

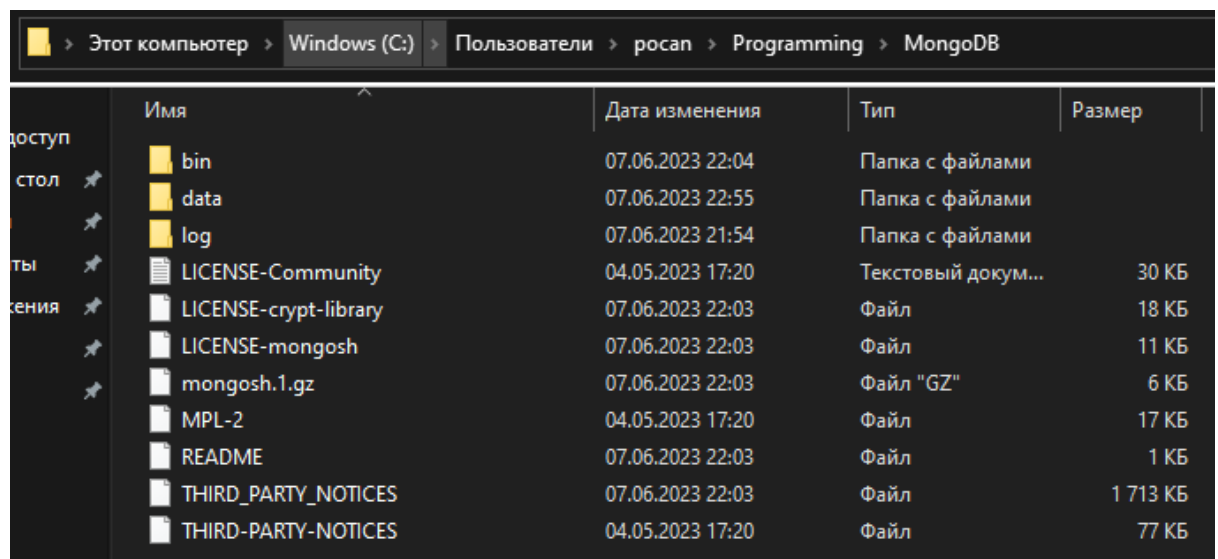
Санкт-Петербург 2022/2023

Цель работы: овладеть практическими навыками установки СУБД MongoDB, работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.

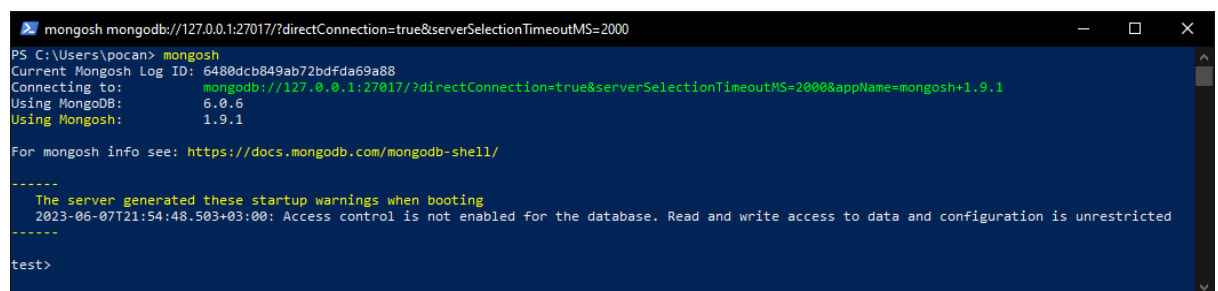
Выполнение:

Практическая работа 5.1

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. db.help()

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
test> db.help()

Database Class:

getMongo           Returns the current database connection
getNames           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 aggregation 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.
getUsers          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.
setUsers          Returns information for all the users in the database.
createCollection  Create new collection
createEncryptedCollection Create new collection with a list of encrypted fields each with unique and auto-created data encryption keys (DEKs). This is a utility function that internally utilizes ClientEncryption.createEncryptedCollection.
createView        Create new view
createRole        Creates a new role.
updateRole        Updates the role's profile on the database on which you run the method. An update to a field completely replaces the previous field's values.
dropRole         Removes the role from the current database.
dropAllRoles     Removes all roles from the current database.
grantRolesToRole  Grants additional roles to a role.
revokeRolesFromRole Removes a one or more roles from a role on the current database.
grantPrivilegesToRole Grants additional privileges to a role.
revokePrivilegesFromRole Removes a one or more privileges from a role on the current database.
getRole          Returns role information for a specified role. Run this method on the role's database. The role must exist on the database on which the method runs.
getRoles         Returns information for all the roles in the database.
currentOp        Runs an aggregation using $currentOp operator. Returns a document that contains information on in-progress operations for the database instance. For further information, see $currentOp.
killOp           Calls the killOp command. Terminates an operation as specified by the operation ID. To find operations and their corresponding IDs, see $currentOp or db.currentOp().
shutdownServer   Calls the shutdown command. Shuts down the current mongod or mongos process cleanly and safely. You must issue the db.shutdownServer() operation against the admin database.
flush          Calls the flush command. Forces the mongod to flush all pending write operations to disk and locks the entire mongod instance to prevent additional writes until the user releases the lock with a corresponding db.flushSyncLock() command.
fsyncLock        Calls the fsyncLock command. Reduces the lock taken by db.fsyncLock() on a mongod instance by 1.
version          returns the db version, uses the buildInfo command
serverStatus     returns the db serverStatus, uses the buildInfo command
lsMaster         Calls the lsMaster command
hello            Calls the hello command
serverBuildInfo  returns the db serverBuildInfo, uses the buildInfo command
serverStatus     returns the server status, uses the serverStatus command
stats            returns the db stats, uses the stats command
hostInfo         Calls the hostInfo command
serverCmdOpts    returns the db serverCmdOpts, uses the getCmdLineOpts command
rotateCertificates Calls the rotateCertificates command
printCollectionStats Prints the collection stats for each collection in the db.
getFreeMonitoringStatus Calls the getFreeMonitoringStatus command
disableFreeMonitoring returns the db disableFreeMonitoring, uses the setFreeMonitoring command
enableFreeMonitoring returns the db enableFreeMonitoring, uses the setFreeMonitoring command
getProfilingStatus returns the db getProfilingStatus, uses the profile command
setProfilingLevel returns the db setProfilingLevel, uses the profile command
setLogLevel      returns the db setLogLevel, uses the setParameter command
getLogComponents returns the db getLogComponents, uses the getParameter command
cloneDatabase    deprecated, non-functional
cloneCollection  deprecated, non-functional
copyDatabase     deprecated, non-functional
cloneShard       returns the db cloneShard, uses the passed in command with help: true
listCommands     Calls the listCommands command
getLastError      Calls the getLastError command
getLastErrorObj  Calls the getLastError command
printShardingStatus Calls sh.status(verbose)
printSecondaryReplicationInfo Prints secondary replication information
```

2. db.help

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
test> db.help

Database Class:

getMongo           Returns the current database connection
getNames           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 aggregation 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.
getUsers          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.
setUsers          Returns information for all the users in the database.
createCollection  Create new collection
createEncryptedCollection Create new collection with a list of encrypted fields each with unique and auto-created data encryption keys (DEKs). This is a utility function that internally utilizes ClientEncryption.createEncryptedCollection.
createView        Create new view
createRole        Creates a new role.
updateRole        Updates the role's profile on the database on which you run the method. An update to a field completely replaces the previous field's values.
dropRole         Removes the role from the current database.
dropAllRoles     Removes all roles from the current database.
grantRolesToRole  Grants additional roles to a role.
revokeRolesFromRole Removes a one or more roles from a role on the current database.
grantPrivilegesToRole Grants additional privileges to a role.
revokePrivilegesFromRole Removes a one or more privileges from a role on the current database.
getRole          Returns role information for a specified role. Run this method on the role's database. The role must exist on the database on which the method runs.
getRoles         Returns information for all the roles in the database.
currentOp        Runs an aggregation using $currentOp operator. Returns a document that contains information on in-progress operations for the database instance. For further information, see $currentOp.
killOp           Calls the killOp command. Terminates an operation as specified by the operation ID. To find operations and their corresponding IDs, see $currentOp or db.currentOp().
shutdownServer   Calls the shutdown command. Shuts down the current mongod or mongos process cleanly and safely. You must issue the db.shutdownServer() operation against the admin database.
flush          Calls the flush command. Forces the mongod to flush all pending write operations to disk and locks the entire mongod instance to prevent additional writes until the user releases the lock with a corresponding db.flushSyncLock() command.
fsyncLock        Calls the fsyncLock command. Reduces the lock taken by db.fsyncLock() on a mongod instance by 1.
version          returns the db version, uses the buildInfo command
serverStatus     returns the db serverStatus, uses the buildInfo command
lsMaster         Calls the lsMaster command
hello            Calls the hello command
serverBuildInfo  returns the db serverBuildInfo, uses the buildInfo command
serverStatus     returns the server status, uses the serverStatus command
stats            returns the db stats, uses the stats command
hostInfo         Calls the hostInfo command
serverCmdOpts    returns the db serverCmdOpts, uses the getCmdLineOpts command
rotateCertificates Calls the rotateCertificates command
printCollectionStats Prints the collection stats for each collection in the db.
getFreeMonitoringStatus Calls the getFreeMonitoringStatus command
disableFreeMonitoring returns the db disableFreeMonitoring, uses the setFreeMonitoring command
enableFreeMonitoring returns the db enableFreeMonitoring, uses the setFreeMonitoring command
getProfilingStatus returns the db getProfilingStatus, uses the profile command
setProfilingLevel returns the db setProfilingLevel, uses the profile command
setLogLevel      returns the db setLogLevel, uses the setParameter command
getLogComponents returns the db getLogComponents, uses the getParameter command
cloneDatabase    deprecated, non-functional
cloneCollection  deprecated, non-functional
copyDatabase     deprecated, non-functional
cloneShard       returns the db cloneShard, uses the passed in command with help: true
listCommands     Calls the listCommands command
getLastError      Calls the getLastError command
getLastErrorObj  Calls the getLastError command
printShardingStatus Calls sh.status(verbose)
printSecondaryReplicationInfo Prints secondary replication information
```

3. db.stats()

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&ser...
{
  db: 'test',
  collections: 0,
  views: 0,
  objects: 0,
  avgObjSize: 0,
  dataSize: 0,
  storageSize: 0,
  indexes: 0,
  indexSize: 0,
  totalSize: 0,
  scaleFactor: 1,
  fsUsedSize: 0,
  fsTotalSize: 0,
  ok: 1
}
test> █
```

4. Создайте БД learn.

5. Получите список доступных БД.

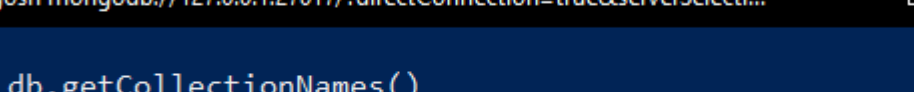
6. Создайте коллекцию unicorns, вставив в нее документ:

```
{name: 'Aurora', gender: 'f', weight: 450}
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serv...
test> use learn
switched to db learn
learn> show dbs
admin    40.00 KiB
config  108.00 KiB
local   40.00 KiB
learn> db.createCollection("unicorns")
{ ok: 1 }
learn> db.unicorns.insertOne({name: 'Aurora', gender: 'f', weight: 450})
{
  acknowledged: true,
  insertedId: ObjectId("6480e59f6e43b6153a1e4340")
}
learn> show dbs
admin    40.00 KiB
config  108.00 KiB
learn    8.00 KiB
local   40.00 KiB
learn>
```

8. Переименуйте коллекцию unicorns.

8. Переименуйте коллекцию unicorns.



The screenshot shows a terminal window with a dark blue background. The title bar at the top is black and contains a green icon, the text "mongosh mongod://127.0.0.1:27017/?directConnection=true&serverSelecti...", and standard window control buttons (minimize, maximize, close). The terminal text is as follows:

```
learn> db.getCollectionNames()
[ 'unicorns' ]
learn> db.unicorns.renameCollection("renamed")
{ ok: 1 }
learn>
```

9. Просмотрите статистику коллекции.

[illegible]

10. Удалите коллекцию.

11. Удалите БД learn.

```
learn> db.renamed.drop()
true
learn> db.createCollection("renamed")
{ ok: 1 }
learn>
```

Практическая работа 5.2

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'});
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

test> use learn
switched to db learn
learn> db.createCollection("unicorns")
{ ok: 1 }
learn> db.unicorns.insertMany([
...   {name: 'Horny', loves: ['carrot', 'papaya'], weight: 600, gender: 'm', vampires: 63},
...   {name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43},
...   {name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182},
...   {name: 'Rooooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99},
...   {name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80},
...   {name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40},
...   {name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39},
...   {name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2},
...   {name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33},
...   {name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54},
...   {name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'},
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6480fe4dcb9cd21efe546bbc"),
    '1': ObjectId("6480fe4dcb9cd21efe546bbd"),
    '2': ObjectId("6480fe4dcb9cd21efe546bbe"),
    '3': ObjectId("6480fe4dcb9cd21efe546bbf"),
    '4': ObjectId("6480fe4dcb9cd21efe546bc0"),
    '5': ObjectId("6480fe4dcb9cd21efe546bc1"),
    '6': ObjectId("6480fe4dcb9cd21efe546bc2"),
    '7': ObjectId("6480fe4dcb9cd21efe546bc3"),
    '8': ObjectId("6480fe4dcb9cd21efe546bc4"),
    '9': ObjectId("6480fe4dcb9cd21efe546bc5"),
    '10': ObjectId("6480fe4dcb9cd21efe546bc6")
  }
}
learn>
```

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.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

learn> db.unicorns.insertOne(
...   {
...     name: 'Dunx',
...     loves: ['grape', 'watermelon'],
...     weight: 704,
...     gender: 'm',
...     vampires: 165
...   }
... )
{
  acknowledged: true,
  insertedId: ObjectId("6480fffdcb9cd21efe546bc7")
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bbc"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bbd"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bbe"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bbf"),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc0"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc1"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc2"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc3"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc4"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc5"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId("6480fffdcb9cd21efe546bc7"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
```

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.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({gender: 'm'}, {_id: 0, name: 1}).sort({name: 1})
[
  { name: 'Dunk' },
  { name: 'Horny' },
  { name: 'Kenny' },
  { name: 'Pilot' },
  { name: 'Raleigh' },
  { name: 'Roooooodies' },
  { name: 'Unicrom' }
]
learn> db.unicorns.find({gender: 'f'}, {_id: 0, name: 1}).sort({name: 1}).limit(3)
[ { name: 'Aurora' }, { name: 'Ayna' }, { name: 'Leia' } ]
learn>
```

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}, {_id: 0}).limit(1)
[
  {
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'}, {_id: 0})
{
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```


Практическое задание 8.1.3:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
[
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bbc"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bbe"),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bbf"),
    name: 'Rooooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc2"),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc3"),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc5"),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId("6480fffdcb9cd21efe546bc7"),
    name: 'Dunk',
    weight: 704,
    vampires: 165
  }
]
learn>
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({}, {_id: 0, name: 1}).sort({$natural: -1})
[
  { name: 'Dunk' },
  { name: 'Nimue' },
  { name: 'Pilot' },
  { name: 'Leia' },
  { name: 'Raleigh' },
  { name: 'Kenny' },
  { name: 'Ayna' },
  { name: 'Solnara' },
  { name: 'Rooooooodles' },
  { name: 'Unicrom' },
  { name: 'Aurora' },
  { name: 'Horny' }
]
learn>
```

Практическое задание 8.1.5:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> 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: 182
  },
  {
    name: 'Roooooodies',
    loves: [ 'apple' ],
    weight: 575,

```

Практическое задание 8.1.6:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({gender: 'f', weight: {$gt: 500, $lt: 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'
  }
]
learn>
```

Практическое задание 8.1.7:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({gender: 'm', weight: {$gt: 500}, loves: {$all: ['grape', 'lemon']}}, {_id: 0})
[
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
learn>
```

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ vampires.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc6"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn>
```

Практическое задание 8.1.9:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({gender: 'm'}, {_id: 0, name: 1, loves: {$slice: 1}}).sort({name: 1})
[
  { name: 'Dunx', loves: [ 'grape' ] },
  { name: 'Horny', loves: [ 'carrot' ] },
  { name: 'Kenny', loves: [ 'grape' ] },
  { name: 'Pilot', loves: [ 'apple' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Roonoodles', loves: [ 'apple' ] },
  { name: 'Unicrom', loves: [ 'energon' ] }
]
learn>
```

Практическое задание 8.2.1:

1. Создайте коллекцию towns, включающую следующие документы:

```
{
  name: "Punxsutawney ",
  populatiuon: 6200,
  last_sensus: ISODate("2008-01-31"),
  famous_for: [""],
  mayor: {
    name: "Jim Wehrle"
  }
},
{
  name: "New York",
  populatiuon: 22200000,
  last_sensus: ISODate("2009-07-31"),
  famous_for: ["status of liberty", "food"],
  mayor: {
    name: "Michael Bloomberg",
    party: "I"
  }
},
{
  name: "Portland",
  populatiuon: 528000,
  last_sensus: ISODate("2009-07-20"),
  famous_for: ["beer", "food"],
  mayor: {
    name: "Sam Adams",
    party: "D"
  }
}
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

learn> db.createCollection("towns")
{ ok: 1 }
learn> db.towns.insertMany([
...   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"
...   }
... })
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("648148c2a9e845da0b593d06"),
    '1': ObjectId("648148c2a9e845da0b593d07"),
    '2': ObjectId("648148c2a9e845da0b593d08")
  }
}
learn>
```

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

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

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

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

Практическое задание 8.2.2:

1. Сформировать функцию для вывода списка самцов единорогов.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

learn> function maleUnicorns() {
...   return db.unicorns.find({gender: 'M'}).toArray();
... };

learn> maleUnicorns().forEach(function(doc) { print(doc.name) })
Horny
Unicrom
Rooooooodles
Kenny
Raleigh
Pilot
Dunx
learn>
```

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

3. Вывести результат, используя forEach.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

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

learn> cursor.forEach(printjson)
{
  _id: ObjectId("6480ffdc9cd21efe546bc7"),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'M',
  vampires: 165
}
{
  _id: ObjectId("6480fe4dcb9cd21efe546bbc"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'M',
  vampires: 63
}
learn>
```

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

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

Практическое задание 8.2.4:

Вывести список предпочтений.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

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

Практическое задание 8.2.5:

Подсчитать количество особей единорогов обоих полов.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.aggregate({$group: {_id: "$gender", count: { $sum: 1 }}})
[ { _id: 'm', count: 7 }, { _id: 'f', count: 5 } ]
learn>
```

Практическое задание 8.2.6:

1. Выполнить команду:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.save({name: 'Barney', loves: ['grape'], weight: 340, gender: 'm'})
TypeError: db.unicorns.save is not a function
learn> db.unicorns.insertOne({name: 'Barney', loves: ['grape'], weight: 340, gender: 'm'});
{ acknowledged: true, insertedId: ObjectId("6481544aa9e845da0b593d09") }
learn>
```

Метод *save* является устаревшим. Вместо него следует использовать либо методы *insertOne*, *insertMany*, либо *updateOne*, *updateMany* и *replaceOne* в зависимости от необходимого случая использования.

2. Проверить содержимое коллекции unicorns.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find().sort({$natural: -1}).limit(1)
[
  {
    _id: ObjectId("6481544aa9e845da0b593d09"),
    name: 'Barney',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
learn>
```

Практическое задание 8.2.7:

1. Для самки единорога Аупа внести изменения в БД: теперь ее вес 800, она убила 51 вампира.

2. Проверить содержимое коллекции unicorns.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

learn> db.unicorns.updateOne({name: 'Ayna'}, {$set: {weight: 800, vampires: 51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Ayna'})
[
  {
    _id: ObjectId("6480fe4dcb9cd21efe546bc1"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'F',
    vampires: 51
  }
]
learn>
```

Практическое задание 8.2.8:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

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

Практическое задание 8.2.9:

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.
2. Проверить содержимое коллекции unicorns.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

learn> db.unicorns.updateMany({gender: 'M'}, {$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
learn> db.unicorns.find({gender: 'M'}, {_id: 0, name: 1, vampires: 1})
[
  { name: 'Horny', vampires: 68 },
  { name: 'Unicrom', vampires: 187 },
  { name: 'Booooooodles', vampires: 104 },
  { name: 'Kenny', vampires: 44 },
  { name: 'Raleigh', vampires: 7 },
  { name: 'Pilot', vampires: 59 },
  { name: 'Dunk', vampires: 170 },
  { name: 'Barney', vampires: 5 }
]
learn>
```


Практическое задание 8.2.10:

1. Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.
2. Проверить содержимое коллекции towns.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.towns.updateOne({name: 'Portland'}, {$unset: {"mayor.party": 1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({name: 'Portland'}, {_id: 0, name: 1, mayor: 1})
[ { name: 'Portland', mayor: { name: 'Sam Adams' } } ]
learn>
```

Практическое задание 8.2.11:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.updateOne({name: 'Pilot'}, {$addToSet: {loves: 'chocolate'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Pilot'}, {_id: 0, name: 1, loves: 1})
[ { name: 'Pilot', loves: [ 'apple', 'watermelon', 'chocolate' ] } ]
learn>
```

Практическое задание 8.2.12:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.updateOne({name: 'Aurora'}, {$addToSet: {loves: {$each: ['sugar', 'lemon']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Aurora'}, {_id: 0, name: 1, loves: 1})
[ { name: 'Aurora', loves: [ 'carrot', 'grape', 'sugar', 'lemon' ] } ]
learn>
```

Практическое задание 8.2.13:

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"
  }
}
```

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 3 }
learn> db.towns.insertMany([
... {
...   name: "Punkstutawney ",
...   popujatiuon: 6200,
...   last_sensus: ISODate("2008-01-31"),
...   famous_for: [ "phil the groundhog" ],
...   mayor: {
...     name: "Tim 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"
...   }
... }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64815e6da9e845da0b593d0a"),
    '1': ObjectId("64815e6da9e845da0b593d0b"),
    '2': ObjectId("64815e6da9e845da0b593d0c")
  }
}
learn>
```

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.towns.deleteMany({'mayor.party': {'$exists': false}})
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find({}, {_id: 0})
[
  {
    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' }
  },
  {
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 2 }
learn> db.towns.find({})
learn>
```

Практическое задание 8.3.1:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.createCollection('habitats')
{ ok: 1 }
learn> db.habitats.insertMany([
...   {
...     _id: "FH",
...     fullName: "Fairy Hollow",
...     description: "A mystical grove filled with whispering willows and sparkling brooks. Known for its abundance of fairy circles,
...   },
...   {
...     _id: "RM",
...     fullName: "Rainbow Meadow",
...     description: "A vibrant meadow with flowers in all colors of the rainbow. During the daytime, rainbows frequently appear, and
...   },
...   {
...     _id: "CE",
...     fullName: "Crystal Enclave",
...     description: "A secluded enclave surrounded by towering crystal formations that shimmer in the sunlight. The crystal formation
...   }
... ])
{
  acknowledged: true,
  insertedIds: { '0': 'FH', '1': 'RM', '2': 'CE' }
}
learn>
```

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.updateOne({name: 'Horny'}, {$set: {habitat: {$ref: 'habitats', $id: 'CE'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne({name: 'Ayna'}, {$set: {habitat: {$ref: 'habitats', $id: 'FH'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne({name: 'Raleigh'}, {$set: {habitat: {$ref: 'habitats', $id: 'RM'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>
```

3. Проверьте содержание коллекции единорогов.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.find({}, {_id: 0, name: 1, habitat: 1})
[
  { name: 'Horny', habitat: DBRef("habitats", 'CE') },
  { name: 'Aurora' },
  { name: 'Unicrow' },
  { name: 'Rooooosodiles' },
  { name: 'Solnara' },
  { name: 'Ayna', habitat: DBRef("habitats", 'FH') },
  { name: 'Kenny' },
  { name: 'Raleigh', habitat: DBRef("habitats", 'RM') },
  { name: 'Leia' },
  { name: 'Pilot' },
  { name: 'Nimue' },
  { name: 'Duns' },
  { name: 'Barry' }
]
learn>
```

Практическое задание 8.3.2:

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

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.createIndex({name: 1}, {unique: true})
name_1
learn>
```

Практическое задание 8.3.3:

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.unicorns.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_', },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
learn> db.unicorns.dropIndex('name_1')
{ nIndexesWas: 2, ok: 1 }
learn> db.unicorns.dropIndex('_id_')
MongoServerError: cannot drop _id index
learn>
```

Практическое задание 8.3.4:

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 (executionTimeMillis)

4. Создайте индекс для ключа value.

5. Получите информацию о всех индексах коллекции numbers.

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?

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
learn> db.createCollection('numbers')
{ ok: 1 }
learn> for(i = 0; i < 100000; i++){db.numbers.insertOne({value: i})}
{ acknowledged: true, insertedId: ObjectId("64819849a9e845da0b5dd0ec") }
learn> db.numbers.find().sort({$natural: -1}).limit(4).explain('executionStats').executionStats.executionTimeMillis
0
learn> db.numbers.find().sort({value: -1}).limit(4).explain('executionStats').executionStats.executionTimeMillis
55
learn> db.numbers.createIndex({value: 1})
value_1
learn> db.numbers.find().sort({value: -1}).limit(4).explain('executionStats').executionStats.executionTimeMillis
5
learn>
```

Заключение:

В ходе лабораторной работы был сделан обзор и практическое применение основных команд MongoDB. Этот опыт обеспечивает необходимую основу для эффективного использования данной СУБД.

Основы структуры MongoDB и её подход к управлению данными были изучены. MongoDB, в отличие от традиционных реляционных баз данных, хранит данные в гибком, JSON-подобном формате.

Были разобраны команды вставки MongoDB для добавления новых документов в коллекцию, включая методы `insertOne`, `insertMany`, и `save`. Были изучены способы чтения данных из MongoDB, включая использование метода `find`, применение фильтров, и методы сортировки и ограничения.

Подходы MongoDB к связыванию данных также были рассмотрены, включая встроенные документы и ссылки. Это дало представление о том, как можно организовать связанные данные в MongoDB.

В результате этой работы получены важные инструменты и знания для эффективного управления данными и создания масштабируемых приложений с использованием MongoDB.