

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
«Работа с БД в СУБД MongoDB»
по дисциплине «Проектирование и реализация баз данных»**

Обучающийся Москалец Данила Алексеевич
Факультет прикладной информатики
Группа K3241
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

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

Цель: овладеть практическими навыками установки СУБД MongoDB.

Ход работы:

1. Устанавливаем MongoDB. Проверяем работоспособность

```
test> db.version()
8.0.9
```

2. db.help(), db.help, db.stats()

```
test> db.help()

Database Class:

getMongo           Returns the current database connection
getName            Returns the name of the DB
getCollectionNames Returns an array containing the names of all collections in the current database.
getCollectionInfos Returns an array of documents with collection information, i.e. collection name and options, for the current database.
runCommand         Runs an arbitrary command on the database.
adminCommand       Runs an arbitrary command against the admin database.
aggregate          Runs a specified admin/diagnostic pipeline which does not require an underlying collection.
getSiblingDB       Returns another database without modifying the db 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 is run.
getUsers           Returns information for all the users in the database.
createCollection   Create new collection
createEncryptedCollection Creates a 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 utilises 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 is run.
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.
fsyncLock          Calls the fsync 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.fsycnunlock() command.
fsyncunlock        Calls the fsyncunlock command. Reduces the lock taken by db.fsycnLock() on a mongod instance by 1.
version            returns the db version. uses the buildInfo command
serverBits         returns the db serverBits. uses the buildInfo command
isMaster           Calls the isMaster command
hello              Calls the hello command
serverBuildInfo    returns the db serverBuildInfo. uses the buildInfo command
serverStatus       returns the server stats. uses the serverStatus command
stats              returns the db stats. uses the dbStats command
hostInfo           Calls the hostInfo command
serverCmdLineOpts  returns the db serverCmdLineOpts. uses the getCmdLineOpts command
rotateCertificates Calls the rotateCertificates command
```

```

test> db.help

Database Class:

getMongo           Returns the current database connection
getName            Returns the name of the DB
getCollectionNames Returns an array containing the names of all collections in the current database.
getCollectionInfos Returns an array of documents with collection information, i.e. collection name and options, for the current database.
runCommand         Runs an arbitrary command on the database.
adminCommand       Runs an arbitrary command against the admin database.
aggregate          Runs a specified admin/diagnostic pipeline which does not require an underlying collection.
getShardingOP      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.
changeUserPassword This includes updates to the user's roles array.
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.
createCollection  Create new collection
createEncryptedCollection Creates a 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 utilises 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.
fsyncLock        Calls the fsync 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.fsycnunlock() command.
fsyncUnlock       Calls the fsyncUnlock 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
isMaster         calls the isMaster command
hello            calls the hello command
serverBuildInfo  returns the db serverBuildInfo. uses the buildInfo command
serverStatus     returns the server stats. uses the serverStatus command
stats            returns the db stats. uses the dbStats command
hostInfo         calls the hostInfo command
serverCmdLineOpts returns the db serverCmdLineOpts. uses the getCmdLineOpts command
rotateCertificates calls the rotateCertificates command
printCollectionStats Prints the collection.stats for each collection in the db.
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

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

```

3. Работа с БД learn и коллекцией unicorns

```

test> use learn
switched to db learn

```

```

learn> show dbs
admin    40.00 KiB
config  92.00 KiB
local   72.00 KiB

```

БД learn ещё не отображается, пока туда не вставить хотя бы один документ.

```
learn> db.unicorns.insertOne({ name: 'Aurora', gender: 'f', weight: 450 })
{
  acknowledged: true,
  insertedId: ObjectId('683236fd2251c04d786c4bd0')
}
```

```
learn> show collections
unicorns
```

Теперь видим learn:

```
learn> show dbs
admin    40.00 KiB
config   92.00 KiB
learn    40.00 KiB
local    72.00 KiB
```

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

```
learn> db.uni.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,
cache_resident=false,checksum=on,colgroups=,collator=,columns=,dictionary=0,encryption=(keyid=,name=),exclusive=false,extract
timestamp=oldest_timestamp,enabled=false,file_metadata=,metadata_file=,panic_corrupt=true,repair=false),internal_item_max=0,int
ey_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,bloom=true,bloom_bit_count=16,bloom
prefix=,start_generation=0,suffix=),merge_max=15,merge_min=0),memory_page_image_max=0,memory_page_max=10m,os_cache_dirty_max=0
_per_child=0,split_pct=90,tiered_storage=(auth_token=,bucket=,bucket_prefix=,cache_directory=,local_retention=300,name=,object
type: 'file',
uri: 'statistics:table:collection-0-13057336992812675018',
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,
  'sleep for LSM checkpoint throttle': 0,
  'sleep for LSM merge throttle': 0,
  'total size of bloom filters': 0
},
autocommit: {
  'retries for readonly operations': 0,
  'retries for update operations': 0
},
backup: {
  'total modified incremental blocks with compressed data': 0,
  'total modified incremental blocks without compressed data': 0
},
'block-manager': {
  'allocations requiring file extension': 4,
  'blocks allocated': 4,
  'blocks freed': 0,
  'checkpoint size': 4096,
  'file allocation unit size': 4096,
  'file bytes available for reuse': 0,
  'file magic number': 120897,
  'file major version number': 1,
  'file size in bytes': 20480,
  'minor version number': 0
},
btree: {
```

...

```

    },
    session: { 'object compaction': 0 },
    transaction: {
      'a reader raced with a prepared transaction commit and skipped an update or updates': 0,
      'number of times overflow removed value is read': 0,
      'race to read prepared update retry': 0,
      'rollback to stable history store keys that would have been swept in non-dryrun mode': 0,
      'rollback to stable history store records with stop timestamps older than newer records': 0,
      'rollback to stable inconsistent checkpoint': 0,
      'rollback to stable keys removed': 0,
      'rollback to stable keys restored': 0,
      'rollback to stable keys that would have been removed in non-dryrun mode': 0,
      'rollback to stable keys that would have been restored in non-dryrun mode': 0,
      'rollback to stable restored tombstones from history store': 0,
      'rollback to stable restored updates from history store': 0,
      'rollback to stable skipping delete rle': 0,
      'rollback to stable skipping stable rle': 0,
      'rollback to stable sweeping history store keys': 0,
      'rollback to stable tombstones from history store that would have been restored in non-dryrun mode': 0,
      'rollback to stable updates from history store that would have been restored in non-dryrun mode': 0,
      'rollback to stable updates removed from history store': 0,
      'rollback to stable updates that would have been removed from history store in non-dryrun mode': 0,
      'update conflicts': 0
    }
  },
  sharded: false,
  size: 65,
  count: 1,
  numOrphanDocs: 0,
  storageSize: 20480,
  totalIndexSize: 20480,
  totalSize: 40960,
  indexSizes: { _id_: 20480 },
  avgObjSize: 65,
  ns: 'learn.uni',
  nindexes: 1,
  scaleFactor: 1
}

```

```

learn> db.uni.drop()
true

```

```

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

```

6.2 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.

Ход работы:

2.1.1:

1. Создайте базу данных *learn*.
2. Заполните коллекцию единорогов *unicorns*
3. Используя второй способ, вставьте в коллекцию единорогов документ
4. Проверьте содержимое коллекции с помощью метода *find*.

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

Заполнение данными:

```
learn> db.unicorns.insert({name: 'Horny', loves: ['carrot', 'papaya'], weight: 600, gender: 'm', vampires: 63});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683241152251c04d786c4bdc') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683241202251c04d786c4bdd') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683241282251c04d786c4bde') }
}
learn>
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683241552251c04d786c4bdf') }
}
learn> db.unicorns.insert({name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6832418e2251c04d786c4be0') }
}
learn> db.unicorns.insert({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683241932251c04d786c4be1') }
}
learn> db.unicorns.insert({name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6832419a2251c04d786c4be2') }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683241a12251c04d786c4be3') }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683241a52251c04d786c4be4') }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683241ac2251c04d786c4be5') }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683241b72251c04d786c4be6') }
}
```

```

learn> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6832431d2251c04d786c4be7') }
}
learn>

```

Проверяем содержимое коллекции:

```

learn> db.unicorns.find()
[
  {
    _id: ObjectId('683241152251c04d786c4bdc'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('683241202251c04d786c4bdd'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683241282251c04d786c4bde'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683241552251c04d786c4bdf'),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6832418e2251c04d786c4be0'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683241932251c04d786c4be1'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    id: ObjectId('6832419a2251c04d786c4be2'),

```

2.2.1:

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

```
learn> db.unicorns.find({gender: 'm'}).limit(3).sort({name: 1})
[
  {
    _id: ObjectId('6832431d2251c04d786c4be7'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('683241152251c04d786c4bdc'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6832419a2251c04d786c4be2'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

```
learn> db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
[
  {
    _id: ObjectId('683241202251c04d786c4bdd'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683241932251c04d786c4be1'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('683241a52251c04d786c4be4'),
    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.

```
learn> db.unicorns.findOne({loves: 'carrot'})
{
  _id: ObjectId('683241152251c04d786c4bdc'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
learn> _
```

2.2.2:

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0}).limit(3).sort({name: 1})
[
  {
    _id: ObjectId('6832431d2251c04d786c4be7'),
    name: 'Dunx',
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('683241152251c04d786c4bdc'),
    name: 'Horny',
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6832419a2251c04d786c4be2'),
    name: 'Kenny',
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).limit(3).sort({name: 1})
[
  {
    _id: ObjectId('6832431d2251c04d786c4be7'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('683241152251c04d786c4bdc'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('6832419a2251c04d786c4be2'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
]
```

2.2.3:

Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('6832431d2251c04d786c4be7'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('683241b72251c04d786c4be6'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('683241152251c04d786c4bdc'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('6832419a2251c04d786c4be2'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
]
```

2.1.4:

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

```
learn> db.unicorns.find({}, {_id: 0, loves: {$slice: 1}})
[
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Aurora',
    loves: [ 'carrot' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    name: 'Unicrom',
    loves: [ 'energgon' ],
    weight: 500,
    gender: 'm',
    vampires: 50
  }
]
```

2.3.1:

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

```
learn> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 700}}, {_id: 0})
[
  {
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

2.3.2:

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

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

2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('683241b72251c04d786c4be6'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

2.3.4:

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

```
learn> db.unicorns.find({gender: 'f'}, {name: 1, loves: {$slice: 1}, _id: 0}).sort({name: 1})
[
  { name: 'Aurora', loves: [ 'carrot' ] },
  { name: 'Ayna', loves: [ 'strawberry' ] },
  { name: 'Leia', loves: [ 'apple' ] },
  { name: 'Nimue', loves: [ 'grape' ] },
  { name: 'Solnara', loves: [ 'apple' ] }
]
```

3.1.1:

1) *Создайте коллекцию towns*

```
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('683255152251c04d786c4be8'),
    '1': ObjectId('683255152251c04d786c4be9'),
    '2': ObjectId('683255152251c04d786c4bea')
  }
}
```

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

```
learn> db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 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.

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

3.1.2

1. Сформировать функцию для вывода списка самцов единорогов.

Ошибка: *MongoInvalidArgumentError: Query filter must be a plain object or ObjectId*

Метод `db.collection.find(fn)` раньше работал в старом MongoDB Shell (mongo), где поддерживался JavaScript-фильтр через `this`. Но я использую новую оболочку **mongosh**, в которой эта возможность отключена из соображений безопасности и совместимости.

Сделаем через курсор:

```
learn> const cursor = db.unicorns.find()
... cursor.forEach(function(unicorn) {
...   if (unicorn.gender === "m") {
...     print(unicorn.name)
...   }
... })
...
Horny
Unicrom
Rooooooodles
Kenny
Raleigh
Pilot
Dunx
```

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

```
learn> var cursor = db.unicorns.find({gender: "m"}).sort({name: 1}).limit(2);
.. null;
```

3. Вывести результат, используя `forEach`.

```
learn> cursor.forEach(function(unicorn) {
...   print(unicorn.name);
... });
Dunx
Horny
```

3.2.1

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

3.2.2

Вывести список предпочтений.

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

3.2.3

Посчитать количество особей единорогов обоих полов.

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

3.3.1

Метод `save()` официально удалён из `mongosh` (начиная с MongoDB 6.x). Раньше он автоматически вставлял или обновлял документ, но теперь его заменили двумя чёткими альтернативами:

`insertOne`, `updateOne`

```
learn> db.unicorns.updateOne(
...   {name: "Barney"},
...   {$set: {loves: ["grape"], weight: 340, gender: "m"}},
...   {upsert: true}
... )
{
  acknowledged: true,
  insertedId: ObjectId('68325cffd3bcbce823d5aa84'),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
```

upsert: true (означает: обновить, если найдено, или вставить новый, если не найден.)

Проверяем:

```
learn> db.unicorns.find({name: "Barney"})
[
  {
    _id: ObjectId('68325cffd3bcbce823d5aa84'),
    name: 'Barney',
    gender: 'm',
    loves: [ 'grape' ],
    weight: 340
  }
]
```

3.3.2

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

Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update(
...   {name: "Ayna"},
...   {$set: {weight: 800, vampires: 51}}
... )
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Метод устарел, лучше использовать updateOne

Проверяем:

```
learn> db.unicorns.find({name: "Ayna"})
[
  {
    _id: ObjectId('683241932251c04d786c4be1'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

3.3.3

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

Проверить содержимое коллекции unicorns.

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

Проверяем:

```
learn> db.unicorns.find({name: "Raleigh"})
[
  {
    _id: ObjectId('683241a12251c04d786c4be3'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

3.3.4

Всем самцам единорогов увеличить количество убитых вампиров на 5.

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

3.3.5

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

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

```
learn> db.towns.find({name: "Portland"})
[
  {
    _id: ObjectId('683255152251c04d786c4bea'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

3.3.6

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

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

3.3.7

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

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

3.4.1

Создайте коллекцию towns, включающую следующие документы:


```
learn> db.towns.find()
[
  {
    _id: ObjectId('683255152251c04d786c4be8'),
    name: 'Punxsutawney',
    populatiuon: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ 'phil the groundhog' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('683255152251c04d786c4be9'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('683255152251c04d786c4bea'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

Удалите документы с беспартийными мэрами.

```
learn> db.towns.remove({"mayor.party": {$exists: false}})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, delete
Many, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
```

Проверьте содержание коллекции.

```
learn> db.towns.find()
[
  {
    _id: ObjectId('683255152251c04d786c4be9'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('683255152251c04d786c4bea'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

Очистите коллекцию.

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

1. Просмотрите список доступных коллекций.

```
learn> show collections
towns
unicorns
learn> db.towns.find()

learn>
```

1. Как используется оператор точка?

Оператор `.` используется для доступа к вложенным полям в документах, например:

```
db.collection.find({"mayor.name": "Sam Adams"})
```

2. Как можно использовать курсор?

Курсор позволяет перебрать документы, полученные через `find()`, с помощью `forEach()` или `while (cursor.hasNext())`.

3. Какие возможности агрегирования данных существуют в MongoDB?

Агрегация выполняется через `aggregate()`, поддерживает `$group`, `$sum`, `$avg`, `$match`, `$sort`, `$count`, `distinct()`, `count()` и др.

4. Какая из функций `save` или `update` более детально позволит настроить редактирование документов коллекции?

`update()` позволяет более точно управлять редактированием, использовать операторы `$set`, `$inc`, `$push`, `$unset` и др.

5. Как происходит удаление документов из коллекции по умолчанию?

Метод `remove({условие})` по умолчанию удаляет все документы, удовлетворяющие условию. Если указать `{justOne: true}` — удалит один.

4.1.1

1. Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.
2. Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания.
3. Проверьте содержание коллекции единорогов.

```
learn> db.habitats.insert({_id: "mountains", name: "Rocky Mountains", description: "High altitude terrain"})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{ acknowledged: true, insertedIds: { '0': 'mountains' } }
learn>
```

```
learn> db.unicorns.update(
...   {name: "Aurora"},
...   {$set: {habitat: {$ref: "habitats", $id: "mountains"}}}
... )
DeprecationWarning: Collection.update() is deprecated. Use updateOne, update
Many, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
learn> db.unicorns.find({'name': 'Aurora'})
[
  {
    _id: ObjectId('683241202251c04d786c4bdd'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef('habitats', 'mountains')
  }
]
```

4.2.1

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

```
learn> db.unicorns.createIndex({name: 1}, {unique: true})
name_1
learn>
```

4.3.1

Получите информацию о всех индексах коллекции `unicorns`.

Удалите все индексы, кроме индекса для идентификатора.

Попытайтесь удалить индекс для идентификатора.

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

```
learn> db.unicorns.dropIndexes()
{
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
```

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

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 *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?

```
learn> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6832732f9765a20da66dd26f') }
}
```

```
learn> db.numbers.find().sort({$natural: -1}).limit(4)
[
  { _id: ObjectId('6832732f9765a20da66dd26f'), value: 99999 },
  { _id: ObjectId('6832732f9765a20da66dd26e'), value: 99998 },
  { _id: ObjectId('6832732f9765a20da66dd26d'), value: 99997 },
  { _id: ObjectId('6832732f9765a20da66dd26c'), value: 99996 }
]
```

```

learn> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: '8F2383EE',
    planCacheShapeHash: '8F2383EE',
    planCacheKey: '7DF350EE',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: { stage: 'COLLSCAN', direction: 'backward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 67,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
      isCached: false,
      stage: 'LIMIT',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      limitAmount: 4,
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 4,
        executionTimeMillisEstimate: 0,
        works: 4,
        advanced: 4,
        needTime: 0,
        needYield: 0,
        saveState: 1,
        restoreState: 1,
        isEOF: 0,
        direction: 'backward',
        docsExamined: 4
      }
    }
  },
  queryShapeHash: '71C04F100DB2038344A2B08A520E31838D2BDF488D60563B2C7886E847B22DE4',
  command: {
    find: 'numbers',
    filter: {},
    sort: { '$natural': -1 },
    limit: 4,
  }
}

```

```

        isEOF: 0,
        direction: 'backward',
        docsExamined: 4
    }
}
},
queryShapeHash: '71C04F100DB2038344A2B08A520E31838D2BDF488D60563B2C7886E847B22DE4',
command: {
  find: 'numbers',
  filter: {},
  sort: { '$natural': -1 },
  limit: 4,
  '$db': 'learn'
},
serverInfo: {
  host: 'DESKTOP-05FGR48',
  port: 27017,
  version: '8.0.9',
  gitVersion: 'f882ef816d531ecfbb593843e4c554fda90ca416'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted',
  internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
ok: 1
}

```

```

learn> db.numbers.createIndex({value: 1})
value_1

```

```

learn> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: '8F2383EE',
    planCacheShapeHash: '8F2383EE',
    planCacheKey: '7DF350EE',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: { stage: 'COLLSCAN', direction: 'backward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
      isCached: false,
      stage: 'LIMIT',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      limitAmount: 4,

```

executionTimeMillis (1): 67

executionTimeMillis (2): 0

Вывод: с индексами поиск идет намного быстрее

1. Назовите способы связывания коллекций в MongoDB:

В MongoDB коллекции можно связывать следующими способами:

Автоматическое связывание через DBRef — используется специальный объект:

{ "\$ref": "collection", "\$id": <значение>, "\$db": <имя_БД> }.

Пример: `company: new DBRef("companies", apple._id).`

Ручное связывание (manual referencing) — сохранение `_id` связанного документа напрямую как значение поля, без использования `DBRef`.

Оба способа позволяют связать документы из разных коллекций, а затем по значению `$id` получить связанные данные через `findOne()`.

2. Сколько индексов можно установить на одну коллекцию в БД MongoDB?

До 64 индексов на одну коллекцию (включая индекс по `_id`).

Как получить информацию о всех индексах базы данных MongoDB?

Для каждой коллекции нужно вызвать:

```
db.<collection>.getIndexes()
```

Пример:

```
db.unicorns.getIndexes()
```

Выводы:

В ходе выполнения лабораторной работы я познакомился с основами работы в СУБД MongoDB. Мне понравилось работать с MongoDB. Всё довольно понятно и удобно - можно просто добавлять документы, не задавая жёсткую структуру.