

**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»

Обучающиеся Савченко Анастасия Сергеевна

Факультет прикладной информатики

Группа K3240

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2023

Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 Выполнение (Часть 1).....	4
1.1 Проверка работоспособности.....	4
2 Выполнение (Часть 2).....	8
Практическое задание 2.1.1.....	8
Практическое задание 2.2.1.....	9
Практическое задание 2.2.2.....	11
Практическое задание 2.1.4.....	13
Практическое задание 2.3.1.....	14
Практическое задание 2.3.2.....	15
Практическое задание 2.3.3.....	16
Практическое задание 2.3.4.....	17
3 Выполнение (Часть 3).....	18
3.1 Практическое задание 3.1.1.....	18
3.2 Практическое задание 3.1.2.....	19
3.3 Практическое задание 3.2.1.....	21
3.4 Практическое задание 3.2.2.....	22
3.5 Практическое задание 3.2.3.....	22
3.6 Практическое задание 3.3.1.....	23
3.7 Практическое задание 3.3.2.....	24
3.8 Практическое задание 3.3.3.....	25
3.9 Практическое задание 3.3.4.....	25
3.10 Практическое задание 3.3.5.....	27
3.11 Практическое задание 3.3.6.....	27
3.12 Практическое задание 3.3.7.....	28
3.13 Практическое задание 3.4.1.....	29
4 Выполнение (Часть 4).....	31
4.1 Практическое задание 4.1.1.....	33
4.2 Практическое задание 4.2.1.....	35
4.3 Практическое задание 4.3.1.....	36
4.4 Практическое задание 4.4.1.....	37
ЗАКЛЮЧЕНИЕ.....	41

ВВЕДЕНИЕ

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

Практическое задание:

Часть 1:

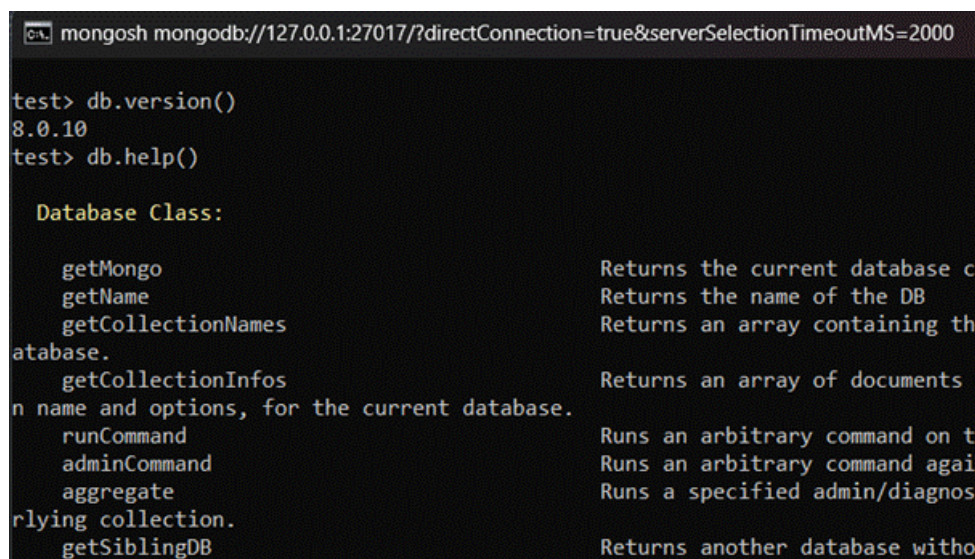
- Установите MongoDB.
- Проверьте работоспособность системы запуском клиента mongo.
- Выполните методы:
 - `db.help()`
 - `db.help`
 - `db.stats()`
- Создайте БД learn.
- Получите список доступных БД.
- Создайте коллекцию unicorns вставив в неё документ {name: 'Aurora', gender: 'f', weight: 450}.
- Просмотрите список текущих коллекций.
- Переименуйте коллекцию unicorns.
- Просмотрите статистику коллекции.
- Удалите коллекцию.
- Удалите БД learn.

1 Выполнение (Часть 1)

1.1 Проверка работоспособности

a) db.help()

Метод db.help() в MongoDB shell (mongosh) используется для вывода справочной информации о доступных методах и командах, которые можно выполнять с текущей базой данных (db) (Рисунок 1).



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

test> db.version()
8.0.10
test> db.help()

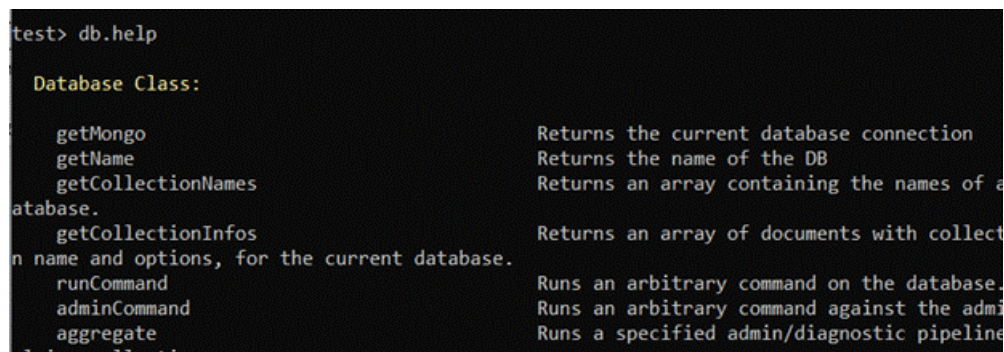
Database Class:

  getMongo           Returns the current database connection
  getName            Returns the name of the DB
  getCollectionNames Returns an array containing the names of the collections in the database.
  getCollectionInfos Returns an array of documents with 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 against the current collection.
  getSiblingDB        Returns another database without creating a new connection.
```

Рисунок 1 – Результат выполнения метода db.help()

b) db.help

В mongosh вывод db.help и db.help() выглядит одинаково, потому что оболочка автоматически выполняет функцию при доступе, если это возможно (рисунок 2).



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

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 the collections in the database.
  getCollectionInfos Returns an array of documents with 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 against the current collection.
  getSiblingDB        Returns another database without creating a new connection.
```

Рисунок 2 – Результат выполнения метода db.help

c) db.stats()

В MongoDB используется для получения статистической информации о текущей базе данных (рисунок 3).

```
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
}
```

Рисунок 3 – Результат выполнения метода db.stats()

Создание базы данных learn

```
test> use learn
switched to db learn
learn> show dbs
admin    40.00 KiB
config  92.00 KiB
local   40.00 KiB
```

Рисунок 4 – Создание базы

База данных learn создана (но не отображается в show dbs до добавления данных).

Требуется создание коллекции unicorns и вставка документа + проверка db.unicorns.find().

```
learn> db.unicorns.insertOne({ name: 'Aurora', gender: 'f', weight: 450 })
{
  acknowledged: true,
  insertedId: ObjectId('684d81da646207232250eb68')
}
```

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('684d7dc1646207232250eb67'),
    name: 'Aurora',
    gender: 'f',
    weight: 450
  }
]
```

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

Рисунок 5 – Создание коллекции и вставка документа

```
learn> show collections
unicorns
```

Рисунок 6 – Просмотр списка коллекций

Переименовали коллекцию unicorns → horses (рисунок 7, 8).

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

Рисунок 7, 8 – Переименование коллекции

Статистику коллекции можно увидеть на рисунке 9.


```
learn> db.stats()
{
  db: 'learn',
  collections: Long('1'),
  views: Long('0'),
  objects: Long('1'),
  avgObjSize: 65,
  dataSize: 65,
  storageSize: 20480,
  indexes: Long('1'),
  indexSize: 20480,
  totalSize: 40960,
  scaleFactor: Long('1'),
  fsUsedSize: 120448815104,
  fsTotalSize: 1023398637568,
  ok: 1
}
```

Рисунок 9 – Статистика коллекции

Удалим коллекцию (рисунок 10).

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

Рисунок 10 – Удаление коллекции и вывод пустого списка коллекций

Удалим БД learn (рисунок 11, 12).

```
learn> use learn
already on db learn
learn> db.dropDatabase()
{ ok: 1, dropped: 'learn' }
```

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

Рисунок 11, 12 – Удаление БД

2 Выполнение (Часть 2)

Практическое задание 2.1.1

Создали базу данных `learn` и заполнили коллекцию единорогов `unicorns` (рисунок 13).

```
learn> show collections

learn> db.unicorns.insert({name: 'Horny', loves: ['carrot', 'papaya'], weight: 600, gender:
... 'm', vampires: 63});
... db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender:
... 'f', vampires: 43});
... db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984,
... gender: 'm', vampires: 182});
... db.unicorns.insert({name: 'Rooooooodles', 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'});
...
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('684d854f646207232250eb7e') }
}
learn> show collections
```

Рисунок 13 – Создание бд и заполнение коллекции способ 1

Используя *второй способ*, вставили в коллекцию единорогов *новый документ* (рисунок 14).

```
learn> doc = { 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(doc)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('684d85b9646207232250eb7f') }
}
```


Рисунок 14 – Вставка нового документа в коллекцию вторым способом

Проверили содержимое коллекции с помощью метода `find` (рисунок 15).

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('684d854f646207232250eb74'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('684d854f646207232250eb75'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('684d854f646207232250eb76'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
]
```

```
learn> db.unicorns.find({name: 'Dunx'})
[
  {
    _id: ObjectId('684d85b9646207232250eb7f'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
learn>
```

Рисунок 15 – Проверка содержимого коллекции

Практическое задание 2.2.1

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

```
db.unicorns.find({gender: 'm'}).sort({name: 1})
```

```
db.unicorns.find({gender: 'f'}).sort({name: 1})
```

На рисунке 16 результат выполнения запроса.

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('684d85b9646207232250eb7f'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('684d854f646207232250eb74'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('684d854f646207232250eb7a'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('684d854f646207232250eb7d'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {

```

Рисунок 16 – Результат выполнения запроса

Ограничьте список самок первыми тремя особями.

```
db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
```

На рисунке 17 результат выполнения запроса.

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('684d854f646207232250eb75'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('684d854f646207232250eb79'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('684d854f646207232250eb7c'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn>
```

Рисунок 17 – Результат выполнения запроса

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

Вариант 1: Использование *findOne*

```
db.unicorns.findOne({ gender: 'f', loves: 'carrot' })
```

```
learn> db.unicorns.findOne({ gender: 'f', loves: 'carrot' })
{
  _id: ObjectId('684d854f646207232250eb75'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```

Вариант 2: Использование *find* + *limit*

```
db.unicorns.find({gender: 'f', loves: 'carrot' }).limit(1)
```

На рисунке 18 результат выполнения запроса.

```
learn> db.unicorns.find({gender: 'f', loves: 'carrot' }).limit(1)
[
  {
    _id: ObjectId('684d854f646207232250eb75'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

Рисунок 18 – Результат выполнения запроса *find* + *limit*

Практическое задание 2.2.2

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

```
db.unicorns.find( {gender: 'm'}, {loves: 0, id: 0} ).sort({name: 1})
```


Пояснение:

- Фильтрация (`{ gender: "m" }`) - Выбираем только единорогов мужского пола

- Проекция (второй аргумент):

`loves: 0` - исключаем поле с предпочтениями

`gender: 0` - исключаем само поле пола (так как мы уже фильтруем по нему)

`_id: 0` - исключаем техническое поле идентификатора

- Сортировка (`.sort({ name: 1 })`) - Сортируем результаты по имени в алфавитном порядке (1 - по возрастанию)

На рисунке 19 результат выполнения запроса.

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

Рисунок 19 – Результат выполнения запроса

Практическое задание 2.2.3

Вывести список единорогов в обратном порядке добавления.

```
db.unicorns.find().sort({ $natural: -1 })
```

На рисунке 20 результат выполнения запроса.


```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('684d85b9646207232250eb7f'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('684d854f646207232250eb7e'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
]
```

Рисунок 20 – Результат выполнения запроса

Практическое задание 2.1.4

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

```
db.unicorns.find(
  {},
  {
    loves: { $slice: 1 },
    _id: 0
  }
)
```

На рисунке 21 результат выполнения запроса.

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

```

Рисунок 21 – Результат выполнения запроса

Практическое задание 2.3.1

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

```
db.unicorns.find(
  {
    gender: "f",
    weight: { $gte: 500, $lte: 700 }
  },
  { _id: 0 }
)
```

На рисунке 22 результат выполнения запроса.

```
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'
  }
]
learn> _
```

Рисунок 22 – Результат выполнения запроса

Практическое задание 2.3.2

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

```
db.unicorns.find(
  {
    gender: "m",
    weight: { $gte: 500 },
    loves: { $all: ["grape", "lemon"] }
  },
  { _id: 0 }
)
```

```
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
  }
]
learn>
```

Рисунок 23 – Результат выполнения запроса

Практическое задание 2.3.3

Найти всех единорогов, не имеющих ключ vampires.

```
db.unicorns.find(
  { vampires: { $exists: false } },
  { _id: 0 }
)
```

```
learn> db.unicorns.find(
...   { vampires: { $exists: false } },
...   { _id: 0 }
... )
[
  {
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn>
```

Рисунок 24 – Результат выполнения запроса

Практическое задание 2.3.4:

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

```
db.unicorns.find(  
  { gender: "m" },  
  { loves: { $slice: 1 }, name: 1, _id: 0 }  
).sort({ name: 1 })
```

```
learn> db.unicorns.find( { gender: "m" }, { loves: { $slice: 1 }, name: 1, _id: 0 } ).sort({ name: 1 })  
[  
  { name: 'Dunx', loves: [ 'grape' ] },  
  { name: 'Horny', loves: [ 'carrot' ] },  
  { name: 'Kenny', loves: [ 'grape' ] },  
  { name: 'Pilot', loves: [ 'apple' ] },  
  { name: 'Raleigh', loves: [ 'apple' ] },  
  { name: 'Rooooooodles', loves: [ 'apple' ] },  
  { name: 'Unicrom', loves: [ 'energon' ] }  
]  
learn> _
```

Рисунок 25 – Результат выполнения запроса

3 Выполнение (Часть 3)

3.1 Практическое задание 3.1.1

- 1) *Создали коллекцию towns* (рисунок 26).

```
learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     population: 6200,
...     last_census: ISODate("2008-01-31"),
...     famous_for: [""],
...     mayor: { name: "Jim Wehrle" }
...   },
...   {
...     name: "New York",
...     population: 22200000,
...     last_census: ISODate("2009-07-31"),
...     famous_for: ["statue of liberty", "food"],
...     mayor: { name: "Michael Bloomberg", party: "I" }
...   },
...   {
...     name: "Portland",
...     population: 528000,
...     last_census: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: { name: "Sam Adams", party: "D" }
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('684db53a0f62299a6150eb67'),
    '1': ObjectId('684db53a0f62299a6150eb68'),
    '2': ObjectId('684db53a0f62299a6150eb69')
  }
}
learn>
```

Рисунок 26 – Созданная коллекция towns

- 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, name: 1, mayor: 1 })
```

На рисунке 27 результат выполнения запроса.

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

Рисунок 27 – Результат выполнения запроса

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

```
db.towns.find(
  { "mayor.party": { $exists: false } },
  {
    _id: 0,
    name: 1,
    mayor: 1
  }
)
```

На рисунке 28 результат выполнения запроса.

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

Рисунок 28 – Результат выполнения запроса.

3.2 Практическое задание 3.1.2

3) Сформировать функцию для вывода списка самцов единорогов.

```
function showMaleUnicorns() {
  var cursor = db.unicorns.find({ gender: "m" }); null;
  cursor.forEach(function(obj) {
    print(obj.name);
  });
}

showMaleUnicorns()
```

На рисунке 29 результат вызова функции.

```

learn> function showMaleUnicorns() {
... var cursor = db.unicorns.find({ gender: "m" }); null;
... cursor.forEach(function(obj) {
...   print(obj.name);
... });
... }
...
[Function: showMaleUnicorns]
learn> showMaleUnicorns()
...
Horny
Unicrom
Roooooodles
Kenny
Raleigh
Pilot
Dunx

```

Рисунок 29 – Результат вызова функции.

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

5) Вывести результат, используя *forEach*.

```

var cursor = db.unicorns.find({ gender: "m" }); null;
cursor.sort({ name: 1 }).limit(2); null;
cursor.forEach(function(obj) {
  print(obj.name);
});

```

На рисунке 30 результат курсора.

```

learn> var cursor = db.unicorns.find({ gender: "m" }); null;
... cursor.sort({ name: 1 }).limit(2); null;
... cursor.forEach(function(obj) {
...   print(obj.name);
... });
...
Dunx
Horny

```

Рисунок 30 – Результат выполнения курсора.

6) Содержание коллекции единорогов unicorns:

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('684d854f646207232250eb74'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('684d854f646207232250eb75'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('684d854f646207232250eb76'),
    name: 'Unicrom',
    loves: [ 'enengon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {

```

3.3 Практическое задание 3.2.1

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Предупреждение об устаревании: функция `Collection.count()` устарела. Используйте `count Documents` или `estimatedDocumentCount`.

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

На рисунке 31 результат выполнения запроса.

```
learn> db.unicorns.count({
...   gender: 'f',
...   weight: { $gte: 500, $lte: 600 }
... })
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
2
learn> db.unicorns.countDocuments({
...   gender: "f",
...   weight: { $gte: 500, $lte: 600 }
... })
...
2
learn>
```

Рисунок 31 – Результат выполнения запроса.

3.4 Практическое задание 3.2.2

Вывести список предпочтений.

```
db.unicorns.distinct("loves")
```

На рисунке 32 результат выполнения запроса.

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

Рисунок 32 – Результат выполнения запроса

3.5 Практическое задание 3.2.3

Подсчитать количество особей единорогов обоих полов.

```
db.unicorns.aggregate([
  { $group: { _id: "$gender", count: { $sum: 1 } } }
])
```

На рисунке 33 результат запроса.

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

Рисунок 33 – Результат выполнения запроса

3.6 Практическое задание 3.3.1:

1. Выполнить команду:

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

2. Проверить содержимое коллекции unicorns.

db.unicorns.find({name: 'Barney'})

На рисунке 34 результат выполнения команды.

```
learn> db.unicorns.save({name: 'Barney', loves: ['grape'],
... weight: 340, gender: 'm'})
TypeError: db.unicorns.save is not a function
learn> db.unicorns.find({name: 'Barney'})
learn> _
```

Рисунок 34 – Результат выполнения команды

В mongosh метод `save()` больше не поддерживается. Можно использовать `insertOne()`.

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

На рисунке 35 результат выполнения команды.

```
learn> db.unicorns.insertOne({ name: 'Barney', loves: ['grape'], weight: 340, gender: 'm' })
{
  acknowledged: true,
  insertedId: ObjectId('684dbfc80f62299a6150eb6a')
}
```

Рисунок 35 – Результат выполнения команды.

Проверить содержимое коллекции unicorns.

db.unicorns.find({ name: 'Barney' }).pretty()

На рисунке 36 результат выполнения команды.

```
learn> db.unicorns.find({name: 'Barney'})
[
  {
    _id: ObjectId('684dbfc80f62299a6150eb6a'),
    name: 'Barney',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
learn> _
```

Рисунок 36 – Результат выполнения команды

3.7 Практическое задание 3.3.2

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

```
db.unicorns.updateOne(
  { name: "Ayna" },
  { $set: { weight: 800, vampires: 51 } }
)
```

На рисунке 37 результат выполнения команд.

```
learn> db.unicorns.updateOne(
...   { name: "Ayna" },
...   { $set: { weight: 800, vampires: 51 } }
... )
...
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
learn>

learn> db.unicorns.find({ name: "Ayna" })
[
  {
    _id: ObjectId('684d854f646207232250eb79'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
learn>
```

Рисунок 37 – Результат выполнения команд

3.8 Практическое задание 3.3.3

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

```
db.unicorns.updateOne(  
  { name: "Raleigh" },  
  { $set: { loves: "redbull" } }  
)
```

Проверить содержимое коллекции: `db.unicorns.find({ name: "Raleigh" })`

На рисунке 38 результат выполнения команд.



```
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: ObjectId('684d854f646207232250eb7b'),  
    name: 'Raleigh',  
    loves: 'redbull',  
    weight: 421,  
    gender: 'm',  
    vampires: 2  
  }  
]  
learn>
```

Рисунок 38 – Результат выполнения команд

3.9 Практическое задание 3.3.4

Всем самцам единорогов увеличить количество убитых вампиров на 5.

```
db.unicorns.updateMany(  
  { gender: "m" },  
  { $inc: { vampires: 5 } }
```


3.10 Практическое задание 3.3.5

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

```
db.towns.updateOne(  
  { name: "Portland" },  
  { $unset: { "mayor.party": "" } }  
)
```

Проверить содержимое коллекции towns.

```
db.towns.find({ name: "Portland" })
```

На рисунке 41 результат выполнения команд.



```
learn> db.towns.updateOne(  
...   { name: "Portland" },  
...   { $unset: { "mayor.party": "" } }  
... )  
...  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}  
learn> db.towns.find({ name: "Portland" })  
[  
  {  
    _id: ObjectId('684db53a0f62299a6150eb69'),  
    name: 'Portland',  
    population: 528000,  
    last_census: ISODate('2009-07-20T00:00:00.000Z'),  
    famous_for: [ 'beer', 'food' ],  
    mayor: { name: 'Sam Adams' }  
  }  
]  
learn>
```

Рисунок 41 – Результат выполнения команд

3.11 Практическое задание 3.3.6

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

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

Проверить содержимое коллекции unicorns.

```
db.unicorns.find({ name: "Pilot" })
```

На рисунке 42 результат выполнения команд.



```
learn> db.unicorns.updateOne(
...   { name: "Pilot" },
...   { $push: { loves: "шоколад" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Pilot" })
...
[
  {
    _id: ObjectId('684d854f646207232250eb7d'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'шоколад' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

Рисунок 42 – Результат выполнения команд

3.12 Практическое задание 3.3.7

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

```
db.unicorns.updateOne(
  { name: "Aurora" },
  { $push: { loves: { $each: ["сахар", "лимоны"] } } }
)
```

Проверить содержимое коллекции: `db.unicorns.find({ name: "Aurora" })`

На рисунке 43 результат выполнения команд.

```

learn> db.unicorns.updateOne(
...   { name: "Aurora" },
...   { $push: { loves: { $each: ["сахар", "лимоны"] } } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Aurora" })
[
  {
    _id: ObjectId('684d854f646207232250eb75'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'сахар', 'лимоны' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]

```

Рисунок 43 – Результат выполнения команд

3.13 Практическое задание 3.4.1

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"}}

```

Удалите документы с беспартийными мэрами.

```

db.towns.deleteMany({

  "mayor.party": { $exists: false }

})

```

Проверим содержание коллекции: db.towns.find()

На рисунке 44 результат выполнения до/после.


```
learn> db.towns.find()
[
  {
    _id: ObjectId('684db53a0f62299a6150eb67'),
    name: 'Punxsutawney',
    population: 6200,
    last_census: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('684db53a0f62299a6150eb68'),
    name: 'New York',
    population: 22200000,
    last_census: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('684db53a0f62299a6150eb69'),
    name: 'Portland',
    population: 528000,
    last_census: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

```
learn> db.towns.deleteMany({
...   "mayor.party": { $exists: false }
... })
...
{ acknowledged: true, deletedCount: 2 }
learn> db.towns.find()
[
  {
    _id: ObjectId('684db53a0f62299a6150eb68'),
    name: 'New York',
    population: 22200000,
    last_census: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
```

Рисунок 44 – Результат выполнения команд

Очистить коллекцию: `db.towns.deleteMany({})`

Просмотреть список доступных коллекций: `show collections`

На рисунке 45 результат выполнения команд.

```
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 3 }
learn> show collections
towns
unicorns
```

Рисунок 45 – Результат выполнения команд

Полное удаление коллекции: `db.towns.drop()`

4 Выполнение (Часть 4)

ТЕОРИЯ:

Автоматическое связывание

1 способ. Используя функциональность DBRef, можно установить автоматическое связывание между документами.

Пример применение данной функциональности: добавить новый документ в коллекцию companies:

```
> apple=({"name": "apple", "year": 1976})
```

```
> db.companies.save(apple)
```

В данном случае сохранение идет с помощью метода save, не insert. Метод save при добавлении нового документа генерирует _id. После сохранения можно вывести документ на консоль: > apple

Далее создать новый документ для коллекции person, у которого ключ company свяжем с только что добавленным документом apple:

```
> steve = ({"name": "Steve", "age": 25, company: new DBRef('companies',  
apple._id)})
```

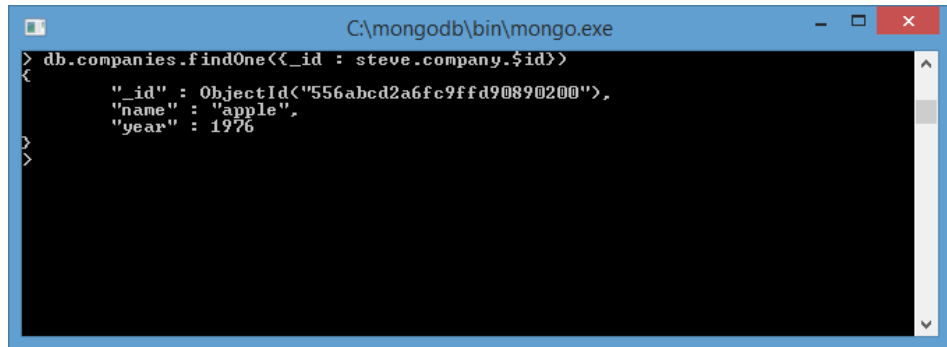
```
> db.users.save(steve)
```



```
C:\mongodb\bin\mongo.exe
> apple=<{"name": "apple", year: 1976}>
< {"name": "apple", "year": 1976 }
> db.companies.save(apple)
WriteResult<< "nInserted" : 1 >>
> apple
<
  "name" : "apple",
  "year" : 1976,
  "_id" : ObjectId<"556abcd2a6fc9ffd90890200">
>
> steve= <{"name": "Steve", age: 25, company: new DBRef<'companies', apple._id>>>
<
  "name" : "Steve",
  "age" : 25,
  "company" : DBRef<"companies", ObjectId<"556abcd2a6fc9ffd90890200">>
>
> db.users.save(steve)
WriteResult<< "nInserted" : 1 >>
>
```

Протестировать:

```
> db.companies.findOne({_id: steve.company.$id})
```

A screenshot of a Windows command prompt window titled "C:\mongodb\bin\mongo.exe". The prompt shows a MongoDB query: `> db.companies.findOne(<<_id : steve.company.$id>>)`. The result is a JSON document: `{ "_id" : ObjectId("556abcd2a6fc9ffd90890200"), "name" : "apple", "year" : 1976 }`.

```
C:\mongodb\bin\mongo.exe
> db.companies.findOne(<<_id : steve.company.$id>>)
{
  "_id" : ObjectId("556abcd2a6fc9ffd90890200"),
  "name" : "apple",
  "year" : 1976
}
```

Посмотрев на примере, можно разобрать организацию ссылок между документами. Для связывания с документом apple использовалось следующее выражение `company: new DBRef('companies', apple._id)`. Формальный синтаксис DBRef следующий:

`{ "$ref" : название_коллекции, "$id": значение [, "$db" : название_бд] }`

При тестировании в качестве запроса на выборку указывается выражение `_id: steve.company.$id`. Так как `person.company` представляет теперь объект `new DBRef('companies', apple._id)`, то нужно конкретизировать параметр `steve.company.$id`.

2 способ. Пусть пользователя нужно связать со страной.

Используется коллекция `countries`, хранящая в качестве идентификатора аббревиатуру и наименование, например:

```
> db.countries.insert({_id:"us", name:"US"})
```

Необходимо добавить в коллекцию пользователей ссылку на страну:

```
>db.users.update({_id:ObjectId("573d7468bfe2c1a9875562e6")},{ $set:
  {country: { $ref:"countries", $id: "us" } } })
```

Теперь можно извлечь из коллекции данные о Томе:

```
>var tom=db.users.findOne({"_id":ObjectId("573d7468bfe2c1a9875562e6")})
```

Можно также получить сведения о стране, в которой находится пользователь, опросив коллекцию стран, передав сохраненный ранее идентификатор `$id`:

```
> db.countries.findOne({_id:tom.country.$id})
```

Можно непосредственно запросить у документа о пользователе имя коллекции, хранящейся в поле ссылки:

```
db[tom.country.$ref].findOne({_id:tom.country.$id})
```

Последние два запроса эквивалентны, но второй в большей степени управляется данными.

4.1 Практическое задание 4.1.1

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

```
db.habitats.insertMany([
  {
    _id: "forest",
    name: "Волшебный лес",
    description: "Густые леса с древними деревьями и чистыми родниками"
  },
  {
    _id: "meadow",
    name: "Цветущий луг",
    description: "Просторные луга с ароматными травами и цветами"
  },
  {
    _id: "mountain",
    name: "Хрустальные горы",
    description: "Скалистые вершины с альпийскими лугами"
  },
  {
```



```

    _id: "cave",
    name: "Тайные пещеры",
    description: "Подземные гроты с сияющими кристаллами"
  }
])

```

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

```

db.unicorns.updateMany(
  { name: { $in: ['Horny', 'Aurora'] } },
  { $set: { habitat: { $ref: "habitats", $id: "forest" } } }
)

```

```

db.unicorns.updateMany(
  { name: { $in: ['Kenny', 'Ayna'] } },
  { $set: { habitat: { $ref: "habitats", $id: "meadow" } } }
)

```

```

db.unicorns.updateMany(
  { name: { $in: ['Unicrom', 'Dunx'] } },
  { $set: { habitat: { $ref: "habitats", $id: "mountain" } } }
)

```

```

db.unicorns.updateOne(
  { name: 'Nimue' },
  { $set: { habitat: { $ref: "habitats", $id: "cave" } } }
)

```

Проверьте содержание коллекции единорогов.

```
db.unicorns.find({ habitat: { $exists: true } })
```

На рисунке 46 результат выполнения команд.

```
learn> db.unicorns.find({ habitat: { $exists: true } })
[
  {
    _id: ObjectId('684d854f646207232250eb74'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    habitat: DBRef('habitats', 'forest')
  },
  {
    _id: ObjectId('684d854f646207232250eb75'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'сахар', 'лимоны' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef('habitats', 'forest')
  },
  {
    _id: ObjectId('684d854f646207232250eb76'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187,
    habitat: DBRef('habitats', 'mountain')
  },
  {
    _id: ObjectId('684d854f646207232250eb79'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51,
    habitat: DBRef('habitats', 'meadow')
  },
  {
    _id: ObjectId('684d854f646207232250eb7a'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 44,
    habitat: DBRef('habitats', 'meadow')
  }
]
```

Рисунок 46 – Результат выполнения команд

4.2 Практическое задание 4.2.1

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

Убедимся, что имена уникальны.

```
db.unicorns.aggregate([
  { $group: { _id: "$name", count: { $sum: 1 } } },
```

```
{ $match: { count: { $gt: 1 } } }  
])
```

Если результат пустой — все имена уникальны, и можно создавать уникальный индекс.

На рисунке 47 результат выполнения команды.

```
learn> db.unicorns.aggregate([  
...   { $group: { _id: "$name", count: { $sum: 1 } } },  
...   { $match: { count: { $gt: 1 } } }  
... ])  
...  
learn> _
```

Рисунок 47 – Результат выполнения команды

Создали уникальные индексы

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

На рисунке 48,49 результат выполнения команды.

```
learn> db.unicorns.ensureIndex({ name: 1 }, { unique: true })  
[ 'name_1' ]  
learn>
```

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

Рисунок 48, 49 – Результат выполнения команды

4.3 Практическое задание 4.3.1

Получите информацию о всех индексах коллекции unicorns.

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

На рисунке 50 результат выполнения команды.

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

Рисунок 50 – Результат выполнения команды

- key — поле(я), по которым построен индекс;
- name — идентификатор индекса (используется при удалении);
- unique: true — признак уникальности (если установлен);

Удалите все индексы, кроме индекса для идентификатора.

```
db.unicorns.dropIndex("name_1")
```

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

На рисунке 51 результат выполнения команд.

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

Рисунок 51 – Результат выполнения команд

Попытайтесь удалить индекс для идентификатора.

```
db.unicorns.dropIndex("_id_")
```

На рисунке 52 результат команды.

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

Рисунок 52 – Результат выполнения команды

Он создаётся автоматически при создании коллекции. Является обязательным и используется для уникальной идентификации каждого документа. Любая попытка удалить его завершится ошибкой.

4.4 Практическое задание 4.4.1

Создайте объемную коллекцию *numbers*, задействовав курсор:

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

DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.

```
const bulk = db.numbers.initializeUnorderedBulkOp();
```

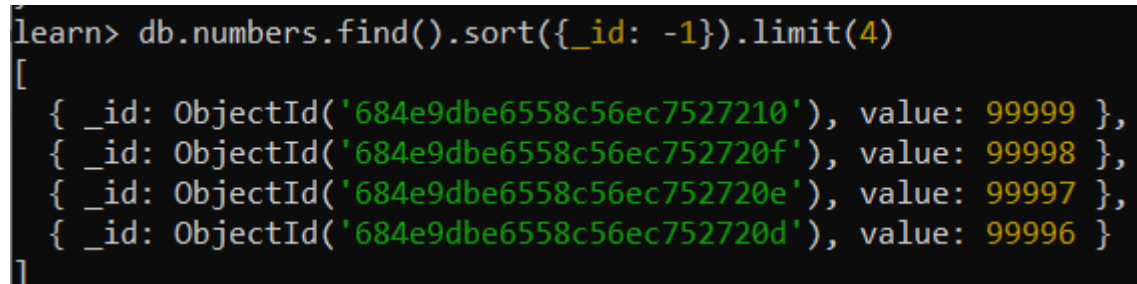
```
for (let i = 0; i < 100000; i++) {
```

```
bulk.insert({ value: i });  
}  
bulk.execute();
```

Выберите последних четыре документа.

```
db.numbers.find().sort({_id: -1}).limit(4)
```

На рисунке 53 результат выполнения команд.



```
learn> db.numbers.find().sort({_id: -1}).limit(4)  
[  
  { _id: ObjectId('684e9dbe6558c56ec7527210'), value: 99999 },  
  { _id: ObjectId('684e9dbe6558c56ec752720f'), value: 99998 },  
  { _id: ObjectId('684e9dbe6558c56ec752720e'), value: 99997 },  
  { _id: ObjectId('684e9dbe6558c56ec752720d'), value: 99996 }  
]
```

Рисунок 53 – Результат выполнения команд

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

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

Ключевые метрики:

-executionTimeMillis: 12 мс

- totalDocsExamined: 4

На рисунке 54 результат выполнения команд.


```

executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 12,
  totalKeysExamined: 4,
  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: 'FETCH',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 4,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 1,
      restoreState: 1,
      isEOF: 0,
      docsExamined: 4,
      alreadyHasObj: 0,
      inputStage: {

```

Рисунок 54 – Результат выполнения команды

Создайте индекс для ключа value.

```
db.numbers.createIndex({ value: 1 })
```

Получите информацию о всех индексах коллекции numbers.

```
db.numbers.getIndexes()
```

На рисунке 55 результат выполнения команд.

```

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

```

Рисунок 55 – Результат выполнения команды

Выполните запрос 2.

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

На рисунке 56 результат выполнения команд.

```
},
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 0,
  totalKeysExamined: 4,
  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,
    inputStage: {
      stage: 'FETCH',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 4,
      advanced: 4,
```

Рисунок 56 – Результат выполнения команды

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

Первый запрос: MongoDB просканировала все 100,000 документов, чтобы выполнить сортировку по value: -1 и вернуть 4 последних.

Второй запрос: Запрос мгновенно вернул 4 записи благодаря использованию индексированной сортировки.

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

1 запрос: 12 мс.

2 запрос: 0 мс.

Запрос с индексом по полю value значительно эффективнее.

Он выполняется в десятки раз быстрее, не требует полной выборки, и не делает сортировку в памяти.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были успешно освоены практические навыки работы с MongoDB, что позволило достичь поставленной цели. В процессе изучения были выполнены следующие ключевые задачи:

- Создание и управление данными:
 - Создана база данных learn и коллекции unicorns, numbers, towns.
 - Освоены методы вставки документов (insert, insertMany, save) и автоматического связывания через DBRef.
- Запросы и выборка данных:
 - Выполнены запросы с фильтрацией по полям, весу, предпочтениям и проверкой наличия ключей.
 - Применены сортировка (sort), ограничение выборки (limit) и проекция (projection).
 - Работа с массивами, включая извлечение элементов через оператор \$slice.
- Обновление и модификация данных:
 - Использованы операторы \$exists, \$all, \$push, \$unset, \$inc для изменения документов.
 - Реализованы методы updateOne и updateMany для обновления данных, включая вложенные массивы.
- Агрегация и анализ данных:
 - Подсчёт документов с помощью count() и агрегация с группировкой (\$group, \$unwind).
 - Создание и тестирование индексов, включая уникальные и составные индексы.
 - Анализ производительности запросов через explain("executionStats"), подтвердивший значительное ускорение при использовании индексов (с 12 мс до 0 мс), за счет исключения полного сканирования коллекции и сортировки в памяти.

Были получены навыки работы с MongoDB, включая манипуляции с данными, создание сложных запросов и оптимизацию производительности через индексы. Работа также продемонстрировала важность правильного проектирования структуры данных и использования индексов для повышения скорости выполнения запросов.