

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

«Национальный исследовательский университет ИТМО»

(Университет ИТМО)

Факультет прикладной информатики

Образовательная программа Мобильные и сетевые технологии

Направление подготовки 09.03.03 Мобильные и сетевые технологии

Дисциплина: Проектирование и реализация баз данных

Практическая работа №6.2

“Работа с БД в СУБД MongoDB”

Обучающийся: Скоблилова Виктория Васильевна К3241

Проверил: Говорова Марина Михайловна

Санкт-Петербург,

2025

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.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 8.0.4 (последняя).

Выполнение:

Практическое задание 2.1.1.

- Создадим базу данных learn и заполним коллекцию unicorns

```
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: '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'});
...
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '_id': ObjectId('68377f1cac02869fda6c4bdb') }
}
learn> |
```

- Вставим в коллекцию unicorns документ с помощью второго способа

```
> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
> db.unicorns.insert(document)
```

```
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: { '_id': ObjectId('68378065ac02869fda6c4bdc') }
}
learn> |
```

3. Используем метод find для проверки содержимого коллекций

```
learn> db.unicorns.find()
[ {
  _id: ObjectId('68377f1cac02869fda6c4bd1'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
},
{
  _id: ObjectId('68377f1cac02869fda6c4bd2'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
  _id: ObjectId('68377f1cac02869fda6c4bd3'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon' ],
  weight: 650,
  gender: 'm',
  vampires: 54
},
{
  _id: ObjectId('68377f1cac02869fda6c4bdb'),
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f'
},
{
  _id: ObjectId('68378065ac02869fda6c4bdc'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
]
learn> |
```

Практическое задание 2.2.1.

- Соформируем запросы для вывода списков самцов и самок, ограничив список первыми тремя особями и отсортировав списки по имени.

Найдем самок, которые любят carrot и ограничим список первой особью, используя функции findOne и limit.

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[ {
  _id: ObjectId('68378065ac02869fda6c4bdc'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
},
{
  _id: ObjectId('68377f1cac02869fda6c4bd1'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
},
{
  _id: ObjectId('68377f1cac02869fda6c4bd7'),
  name: 'Kenny',
  loves: [ 'grape', 'lemon' ],
  weight: 690,
  gender: 'm',
  vampires: 39
},
{
  _id: ObjectId('68377f1cac02869fda6c4bda'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon' ],
  weight: 650,
  gender: 'm',
  vampires: 54
},
]

learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[ {
  _id: ObjectId('68377f1cac02869fda6c4bd2'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
  _id: ObjectId('68377f1cac02869fda6c4bd6'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 733,
  gender: 'f',
  vampires: 40
},
{
  _id: ObjectId('68377f1cac02869fda6c4bd9'),
  name: 'Leia',
  loves: [ 'apple', 'watermelon' ],
  weight: 601,
  gender: 'f',
  vampires: 33
}
]
```

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})  
{  
  _id: ObjectId('68377f1cac02869fda6c4bd2'),  
  name: 'Aurora',  
  loves: [ 'carrot', 'grape' ],  
  weight: 450,  
  gender: 'f',  
  vampires: 43  
}  
learn> |  
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)  
[  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd2'),  
    name: 'Aurora',  
    loves: [ 'carrot', 'grape' ],  
    weight: 450,  
    gender: 'f',  
    vampires: 43  
}  
]  
learn> |
```

Практическое задание 2.2.2:

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

Практическое задание 2.2.3:

Выведем список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({$natural: -1})
[ {
    _id: ObjectId('68378065ac02869fda6c4bdc'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
},
{
    _id: ObjectId('68377f1cac02869fda6c4bdb'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
},
{
    _id: ObjectId('68377f1cac02869fda6c4bda'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
},
{
    _id: ObjectId('68377f1cac02869fda6c4bd9'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
},
```

Практическое задание 2.1.4:

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

```
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
},
{
    name: 'Unicrom',
    loves: [ 'energon' ],
    weight: 984,
    gender: 'm',
    vampires: 182
},
{
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
},
```

Практическое задание 2.3.1:

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

```
learn> db.unicorns.find({gender: 'f', weight: {$lt : 700}}, {_id: 0})
[
  {
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    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 2.3.2:

Выведем список 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
  }
]
learn> |
```

Практическое задание 2.3.3:

Найдем всех единорогов, не имеющих ключа vampires.

```
learn> db.unicorns.find({vampires: {$exists : false}})
[  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bdb'),  
    name: 'Nimue',  
    loves: [ 'grape', 'carrot' ],  
    weight: 540,  
    gender: 'f'  
  }  
]  
learn> |
```

Практическое задание 2.3.4:

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

```
learn> db.unicorns.find({gender: 'm'}, {name: 1, loves: {$slice: 1}}).sort({name: 1})
[  
  {  
    _id: ObjectId('68378065ac02869fda6c4bdc'),  
    name: 'Dunx',  
    loves: [ 'grape' ]  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd1'),  
    name: 'Horny',  
    loves: [ 'carrot' ]  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd7'),  
    name: 'Kenny',  
    loves: [ 'grape' ]  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bda'),  
    name: 'Pilot',  
    loves: [ 'apple' ]  
  },  
]
```

Практическое задание 3.1.1:

Вставим новые документы в коллекцию towns:

```
learn> db.towns.find()
[  
  {  
    _id: ObjectId('68379f3553643ce32e6c4bd0'),  
    name: 'Punxsutawney ',  
    population: 6200,  
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),  
    famous_for: [ '' ],  
    mayor: { name: 'Jim Wehrle' }  
  },  
  {  
    _id: ObjectId('68379f7653643ce32e6c4bd1'),  
    name: 'New York',  
    population: 22200000,  
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),  
    famous_for: [ 'status of liberty', 'food' ],  
    mayor: { name: 'Michael Bloomberg', party: 'I' }  
  },  
  {  
    _id: ObjectId('68379f8753643ce32e6c4bd2'),  
    name: 'Portland',  
    population: 528000,  
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),  
    famous_for: [ 'beer', 'food' ],  
    mayor: { name: 'Sam Adams', party: 'D' }  
  }  
]
```

Сформируем запрос, который возвращает список городов с независимыми мэрами (party="I") и выведем 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' }  
  }  
]  
learn> |
```

Далее сформируем запрос, который возвращает список беспартийных мэров (party отсутствует) и выведем 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' } } ]
```

Практическое задание 3.1.2:

Сформируем функцию для вывода списка самцов единорогов.

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

Выведем результат, используя forEach.

```
learn> male_uni_list = function() {return {gender: 'm'}}; null;
null
learn> var cursor = db.unicorns.find(male_uni_list()).sort({name: 1}).limit(2); null;
null
learn> cursor.forEach(function (obj) { print(obj); })
{
  _id: ObjectId('68378065ac02869fda6c4bdc'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('68377f1cac02869fda6c4bd1'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

Практическое задание 3.2.1:

Выведем количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 3.2.2:

Выведем список предпочтений.

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

Практическое задание 3.2.3:

Подсчитаем количество особей единорогов обоих полов.

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

Практическое задание 3.3.1:

1)Выполним команду:

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

2)Проверим содержимое коллекции unicorns.

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

Новый документ вставляем методом `insert`, так как не можем воспользоваться командой `save()`. В связи с упрощением API данная команда не поддерживается в Mongo Shell и удалена в MongoDB.

Практическое задание 3.3.2:

Внесем изменения в БД для самки единорога Ayna : теперь ее вес 800, она убила 51 вампира.

Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update({name: 'Ayna'}, {$set: {weight: 800, vampires: 51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Ayna'})
[
  {
    _id: ObjectId('68377f1cac02869fda6c4bd6'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

Практическое задание 3.3.3:

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

Проверяем содержимое коллекции unicorns.

```
learn> db.unicorns.update({name: 'Raleigh'}, {$set: {loves: ['redbull']}})  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}  
learn> db.unicorns.find({name: 'Raleigh'})  
[  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd8'),  
    name: 'Raleigh',  
    loves: [ 'redbull' ],  
    weight: 421,  
    gender: 'm',  
    vampires: 2  
  }  
]  
learn> |
```

Практическое задание 3.3.4:

Всем самцам единорогов увеличиваем количество убитых вампиров на 5.
Проверяем содержимое коллекции unicorns.

```
learn> db.unicorns.update({gender: 'm'}, {$inc: {vampires: 5}}, {multi: true})  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 8,  
  modifiedCount: 8,  
  upsertedCount: 0  
}  
learn> db.unicorns.find({gender: 'm'})  
[  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd1'),  
    name: 'Horny',  
    loves: [ 'carrot', 'papaya' ],  
    weight: 600,  
    gender: 'm',  
    vampires: 68  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd3'),  
    name: 'Unicorn',  
    loves: [ 'energon', 'redbull' ],  
    weight: 984,  
    gender: 'm',  
    vampires: 187  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd4'),  
    name: 'Unicorn',  
    loves: [ 'energon', 'redbull' ],  
    weight: 984,  
    gender: 'm',  
    vampires: 187  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd5'),  
    name: 'Unicorn',  
    loves: [ 'energon', 'redbull' ],  
    weight: 984,  
    gender: 'm',  
    vampires: 187  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd6'),  
    name: 'Unicorn',  
    loves: [ 'energon', 'redbull' ],  
    weight: 984,  
    gender: 'm',  
    vampires: 187  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd7'),  
    name: 'Unicorn',  
    loves: [ 'energon', 'redbull' ],  
    weight: 984,  
    gender: 'm',  
    vampires: 187  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd8'),  
    name: 'Raleigh',  
    loves: [ 'redbull' ],  
    weight: 421,  
    gender: 'm',  
    vampires: 2  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd9'),  
    name: 'Unicorn',  
    loves: [ 'energon', 'redbull' ],  
    weight: 984,  
    gender: 'm',  
    vampires: 187  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd0'),  
    name: 'Unicorn',  
    loves: [ 'energon', 'redbull' ],  
    weight: 984,  
    gender: 'm',  
    vampires: 187  
  },  
  {  
    _id: ObjectId('68377f1cac02869fda6c4bd1'),  
    name: 'Horny',  
    loves: [ 'carrot', 'papaya' ],  
    weight: 600,  
    gender: 'm',  
    vampires: 68  
  }  
]
```

Практическое задание 3.3.5:

Изменим информацию о городе Портланд: мэр этого города теперь беспартийный.
Проверяем содержимое коллекции towns.

```
learn> db.towns.update({name: 'Portland'}, {$unset: {'mayor.party': 1}})  
learn> db.towns.find().sort({$natural: -1})  
[  
  {  
    _id: ObjectId('68379f8753643ce32e6c4bd2'),  
    name: 'Portland',  
    population: 528000,  
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),  
    famous_for: [ 'beer', 'food' ],  
    mayor: { name: 'Sam Adams' }  
  },  
  {  
    _id: ObjectId('68379f87653643ce32e6c4bd1')  
  }  
]
```

Практическое задание 3.3.6:

Изменим информацию о самце единорога Pilot: теперь он любит и шоколад. Проверим содержимое коллекции unicorns.

```
learn> db.unicorns.update({name: 'Pilot'}, {$push: {loves: 'chocolate'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Pilot'}, {})
[
  {
    _id: ObjectId('68377f1cac02869fda6c4bda'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
learn> |
```

Практическое задание 3.3.7:

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

Проверим содержимое коллекции unicorns.

```
learn> db.unicorns.update({name: 'Aurora'}, {$push: {loves: {$each: ['sugar', 'lemon']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Aurora'}, {})
[
  {
    _id: ObjectId('68377f1cac02869fda6c4bd2'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn> |
```

Практическое задание 3.4.1:

Удалим документы с беспартийными мэрами.

Проверим содержание коллекции и очистим коллекцию.

Просмотрим список доступных коллекций.

```

learn> db.towns.remove({'mayor.party': {$exists: false}})
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId('6837b49d53643ce32e6c4bd6'),
    name: 'New York',
    population: 2200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('6837b4af53643ce32e6c4bd7'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn> |
learn> db.towns.deleteMany({});
{ acknowledged: true, deletedCount: 2 }
learn> show collections
towns
unicorns
learn> |

```

Практическое задание 4.1.1:

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

```

learn> db.habitats.insertMany([
...   {
...     _id: "meadow",
...     fullName: "Sunny Meadow",
...     description: "A wide, sunny meadow with flowers where unicorns love to graze."
...   },
...   {
...     _id: "forest",
...     fullName: "Mystic Forest",
...     description: "A dense forest with tall trees, perfect for unicorns to hide and play."
...   },
...   {
...     _id: "mountain",
...     fullName: "Crystal Mountain",
...     description: "High mountains with clear air and sparkling stones, a rare but magical unicorn habitat."
...   }
... ]);
...
{
  acknowledged: true,
  insertedIds: { '0': 'meadow', '1': 'forest', '2': 'mountain' }
}
learn> |

```

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

Проверим содержание коллекции единорогов.

```
learn> db.unicorns.update({name: 'Aurora'}, {$set: {habitat: {$ref: 'habitats', $id: 'meadow'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.update({name: 'Unicrom'}, {$set: {habitat: {$ref: 'habitats', $id: 'forest'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.update({name: 'Horny'}, {$set: {habitat: {$ref: 'habitats', $id: 'mountain'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

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

```
[{
  "_id": ObjectId('6837b93153643ce32e6c4bd9'),
  "name": 'Horny',
  "loves": [ 'carrot', 'papaya' ],
  "weight": 600,
  "gender": 'm',
  "vampires": 63,
  "habitat": DBRef('habitats', 'mountain')
},
{
  "_id": ObjectId('6837b93153643ce32e6c4bda'),
  "name": 'Aurora',
  "loves": [ 'carrot', 'grape' ],
  "weight": 450,
  "gender": 'f',
  "vampires": 43,
  "habitat": DBRef('habitats', 'meadow')
},
{
  "_id": ObjectId('6837b93153643ce32e6c4bdb'),
  "name": 'Unicrom',
  "loves": [ 'energon', 'redbull' ],
  "weight": 984,
  "gender": 'm',
  "vampires": 182,
  "habitat": DBRef('habitats', 'forest')
}]
```

Практическое задание 4.2.1:

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

Все имена не пустые и уникальные, следовательно, можно задать вот это
`db.unicorns.createIndex({ "name": 1 }, {"unique": true})`

Пример:

```
learn> db.unicorns.createIndex({{"name": 1}, {"unique": true}}
name_1
learn> db.unicorns.insert({name: 'Horny', dob: new Date(1992,2,13,7,47), loves: ['carrot','chocolate'], weight: 600, gender: 'm', vampires: 63});
Uncatched:
MongoBulkWriteError: E11000 duplicate key error collection: learn.unicorns index: name_1 dup key: { name: "Horny" }
Result: BulkWriteResult {
  insertedCount: 0,
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {},
  insertedIds: {}
}
Write Errors: [
  WriteError {
    err: {
      index: 0,
      code: 11000,
      errmsg: 'E11000 duplicate key error collection: learn.unicorns index: name_1 dup key: { name: "Horny" }',
      errInfo: undefined,
      op: {
        name: 'Horny',
        dob: ISODate('1992-03-13T04:47:00.000Z'),
        loves: ['carrot', 'chocolate'],
        weight: 600,
        gender: 'm',
        vampires: 63,
        _id: ObjectId('6837bcee53643ce32e6c4bf1')
      }
    }
  }
]
learn> |
```

Практическое задание 4.3.1:

Получим информацию о всех индексах коллекции unicorns.

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

Удалим все индексы, кроме индекса для идентификатора.

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

learn> |
```

Попытаемся удалить индекс для идентификатора.

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

Практическое задание 4.4.1:

Создадим объемную коллекцию numbers, задействовав курсор:

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

```
learn> for(i = 0; i < 100000; i++) {  
...     db.numbers.insert({value: i})  
... }  
...  
  
{  
    acknowledged: true,  
    insertedIds: { '0': ObjectId('6837beae53643ce32e6e199c') }  
}  
learn>  
learn>  
learn> |
```

Выберем последних четырех документа.

```
learn> db.numbers.find().sort({value: -1}).limit(4)  
[  
    { _id: ObjectId('6837beae53643ce32e6e199c'), value: 99999 },  
    { _id: ObjectId('6837beae53643ce32e6e199b'), value: 99998 },  
    { _id: ObjectId('6837beae53643ce32e6e199a'), value: 99997 },  
    { _id: ObjectId('6837beae53643ce32e6e1999'), value: 99996 }  
]  
learn> |
```

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

```
learn> db.numbers.find().sort({value: -1}).limit(4).explain("executionStats")  
{  
    explainVersion: '1',  
    queryPlanner: {  
        namespace: 'learn.numbers',  
        parsedQuery: {},  
        indexFilterSet: false,  
        queryHash: 'BA27D965',  
        planCacheShapeHash: 'BA27D965',  
        planCacheKey: '7A892B81',  
        optimizationTimeMillis: 0,  
        maxIndexedOrSolutionsReached: false,  
        maxIndexedAndSolutionsReached: false,  
        maxScansToExplodeReached: false,  
        prunedSimilarIndexes: false,  
        winningPlan: {  
            isCached: false,  
            stage: 'SORT',  
            sortPattern: { value: -1 },  
            memLimit: 104857600,  
            limitAmount: 4,  
            type: 'simple',  
            inputStage: { stage: 'COLLSCAN', direction: 'forward' }  
        },  
        rejectedPlans: []  
    },  
    executionStats: {  
        executionSuccess: true,  
        nReturned: 4,  
        executionTimeMillis: 42,  
        totalKeysExamined: 0,  
        totalDocsExamined: 118187,  
        executionStages: {  
            isCached: false,  
            stage: 'SORT',  
            nReturned: 4,  
            executionTimeMillisEstimate: 32,  
            works: 118193,  
            advanced: 4,  
            needTime: 118188,  
            needYield: 0,  
            numYields: 0,  
            totalLength: 118193  
        }  
    }  
}
```

executionTimeMillis: 42

Создадим индекс для ключа value.

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

Получим информацию о всех индексах коллекции numbers.

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

Выполним запрос 2.

```
learn> db.numbers.find().sort({value: -1}).limit(4).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'BA27D965',
    planCacheShapeHash: 'BA27D965',
    planCacheKey: '7A892B81',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: {
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { value: 1 },
          indexName: 'value_1',
          isMultiKey: false,
          multiKeyPaths: { value: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'backward',
          indexBounds: { value: '[MaxKey, MinKey]' }
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 10,
```

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

executionTimeMillis: 10

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

Запрос без индексов исполнялся в 4 раза дольше

Вывод по проделанной работе:

В процессе выполнения лабораторной работы я овладела 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.