Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «Работа с БД в СУБД MongoDB»

по дисциплине «**Проектирование и реализация баз данных**»

Вариант 5

Автор: Логачев Даниил

Факультет: ИКТ

Группа: К32402

Преподаватель: Говорова М. М.

Дата: 25.09.2023

Санкт-Петербург 2023

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.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 6.0.6 (текущая).

Выполнение:

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

Задание 8.1.1:

1. Создайте базу данных learn.

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

2. Заполните коллекцию единорогов unicorns:

```
learn> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647785cd5500874ee8159151") }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647785cd5500874ee8159152") }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647785cd5500874ee8159153") }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647785cd5500874ee8159154") }
}
learn> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647785cd5500874ee8159155") }
}
learn> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647785cd5500874ee8159156") }
}
learn> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647785cd5500874ee8159157") }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647785cd5500874ee8159158") }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647785cd5500874ee8159159") }
}
```

```
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647786055500874ee815915a") }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647786055500874ee815915b") }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("647786085500874ee815915c") }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477860d5500874ee815915d") }
}
```

3. Используя второй способ, вставьте в коллекцию единорогов документ:

```
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("647789f15500874ee815915e") }
}
```

4. Проверьте содержимое коллекции с помощью метода find.

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId("647785cd5500874ee8159151"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("647785cd5500874ee8159152"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("647785cd5500874ee8159153"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("647785cd5500874ee8159154"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("647785cd5500874ee8159155"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("647785cd5500874ee8159156"),
    name: 'Ayna',
```

Задание 8.1.2:

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("647789f15500874ee815915e"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("647785cd5500874ee8159151"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("647785cd5500874ee8159157"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

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

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId("647785cd5500874ee8159152"),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId("647785cd5500874ee8159152"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Задание 8.1.3:
Модифицируйте запрос для вывода списков самцов единорогов, исключив из результата информацию о предпочтениях и поле.

```
learn>  db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
[
  {
    _id: ObjectId("647785cd5500874ee8159151"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId("647785cd5500874ee8159153"),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId("647785cd5500874ee8159154"),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId("647785cd5500874ee8159157"),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId("647785cd5500874ee8159158"),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId("647786055500874ee815915a"),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId("647786085500874ee815915c"),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
```

Задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId("647789f15500874ee815915e"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("6477860d5500874ee815915d"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("647786085500874ee815915c"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId("647786055500874ee815915b"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("647786055500874ee815915a"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId("647785cd5500874ee8159159"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
```

Задание 8.1.5:
Вывести 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
  },
  {
    name: 'Solnara',
    loves: [ 'apple' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    name: 'Ayna',
    loves: [ 'strawberry' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    name: 'Kenny',
```

Задание 8.1.6:
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'
  },
  {
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Задание 8.1.7:

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
  }
]
```

Задание 8.1.8:

Найти всех единорогов, не имеющих ключ vampires.

```
learn> db.unicorns.find({vampires: {$exists: 0}})
[
  {
    _id: ObjectId("647786055500874ee815915b"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("6477860d5500874ee815915d"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Задание 8.1.9:

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

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

8.2 ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB.

ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ

Задание 8.2.1:

1. Создайте коллекцию towns, включающую следующие документы:

```
learn> db.towns.find()
[
  {
    _id: ObjectId("64779bf85500874ee815915f"),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId("64779c145500874ee8159160"),
    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("64779c205500874ee8159161"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

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: 0}}, {name: 1, mayor: 1, _id: 0})
[ { name: 'Punxsutawney ', mayor: { name: 'Jim Wehrle' } } ]
learn>
```

Задание 8.2.2:

1. Сформировать функцию для вывода списка самцов единорогов.

```
learn> get_male = function() {return this.gender == 'm'}
[Function: get_male]
```

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

3. Вывести результат, используя forEach.

```
learn> var cursor = db.unicorns.find({$where: get_male}).limit(2).sort({name: 1})

learn> cursor.forEach(obj => {print(obj.name)})
Dunx
Horny
```

4. Содержание коллекции единорогов unicorns:

```
db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight:
600, gender: 'm', vampires: 63});

db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight:
450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'],
weight: 984, gender: 'm', vampires: 182});

db.unicorns.insert({name: 'Roooooodles', 44), 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'});

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

Задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
learn> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 600}}).count()
3
```

Задание 8.2.4:

Вывести список предпочтений.

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

Задание 8.2.5:

Подсчитать количество особей единорогов обоих полов.

```
learn> db.unicorns.aggregate({$group: {_id: '$gender', count: {$sum: 1}}})
[ { _id: 'm', count: 8 }, { _id: 'f', count: 6 } ]
```

Задание 8.2.6:

1. Выполнить команду:

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

```
learn> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedId: ObjectId("6477a2a35500874ee8159162")
}
```

2. Проверить содержимое коллекции unicorns.

```
  {
    _id: ObjectId("6477a2a35500874ee8159162"),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
```

Задание 8.2.7:

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

2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.updateOne({name: 'Ayna'}, {$set: {name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 800, gender: 'f', vampires: 51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Ayna'})
[
  {
    _id: ObjectId("647785cd5500874ee8159156"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

Задание 8.2.8:

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

2. Проверить содержимое коллекции unicorns.

```
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("647785cd5500874ee8159158"),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Практическое задание 8.2.9:

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.

2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update({gender: 'm'}, {$inc: {vampires: 5}}, {multi: 1})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 9,
  modifiedCount: 9,
  upsertedCount: 0
}
learn> db.unicorns.find({gender: 'm'})
[
  {
    _id: ObjectId("647785cd5500874ee8159151"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId("647785cd5500874ee8159153"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId("647785cd5500874ee8159154"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104
  },
  {
    _id: ObjectId("647785cd5500874ee8159157"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 44
  },
  {
    _id: ObjectId("647785cd5500874ee8159158"),
```

Практическое задание 8.2.10:

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

2. Проверить содержимое коллекции towns.

```
learn> db.towns.updateOne({name: 'Portland'}, {$unset: {'mayor.party': 1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({name: 'Portland'})
[
  {
    _id: ObjectId("64779c205500874ee8159161"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

Практическое задание 8.2.11:

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

2. Проверить 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("647786055500874ee815915a"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  },
```

Практическое задание 8.2.12:

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

2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update({name: 'Aurora'}, {$addToSet: {loves: {$each: ['sugar', 'lemon']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Aurora'})
[
  {
    _id: ObjectId("647785cd5500874ee8159152"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 8.2.13:

1. Создайте коллекцию towns, включающую следующие документы:

```
{name: "Punxsutawney ",
popujatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: ["phil the groundhog"],
mayor: {
   name: "Jim Wehrle"
   }}

{name: "New York",
popujatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
   name: "Michael Bloomberg",
   party: "I"}}

{name: "Portland",
popujatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
   name: "Sam Adams",
   party: "D"}}
```

```
learn> db.towns.insert({name: "Punxsutawney ",
... popujatiuon: 6200,
... last_sensus: ISODate("2008-01-31"),
... famous_for: ["phil the groundhog"],
... mayor: {
...     name: "Jim Wehrle"
... }}
... )
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a71c5500874ee8159163") }
}
learn> db.towns.insert({name: "New York",
... popujatiuon: 22200000,
... last_sensus: ISODate("2009-07-31"),
... famous_for: ["status of liberty", "food"],
... mayor: {
...     name: "Michael Bloomberg",
... party: "I"}}
... )
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a7315500874ee8159164") }
}
learn> db.towns.insert(
... {name: "Portland",
... popujatiuon: 528000,
... last_sensus: ISODate("2009-07-20"),
... famous_for: ["beer", "food"],
... mayor: {
...     name: "Sam Adams",
... party: "D"}}
... )
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a7415500874ee8159165") }
}
```

2. Удалите документы с беспартийными мэрами.

```
learn> db.towns.remove({'mayor.party': {$exists: 0}})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 3 }
```

3. Проверьте содержание коллекции.

4. Очистите коллекцию.

5. Просмотрите список доступных коллекций.

```
learn> db.towns.drop()
true
learn> show collections
unicorns
learn>
```

8.3 ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

Задание 8.3.1:

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

```
learn> db.unicorn_habitats.insertMany([
...   {
...     "zone_id": 1,
...     "zone_name": "Elven Forest",
...     "zone_description": "Elven Forest - Home of the Unicorns"
...   },
...   {
...     "zone_id": 2,
...     "zone_name": "Magical Plains",
...     "zone_description": "Magical Plains - Haven of the Unicorns"
...   },
...   {
...     "zone_id": 3,
...     "zone_name": "Celestial Meadows",
...     "zone_description": "Celestial Meadows - Enchanting Dwelling of the Unicorns"
...   },
...   {
...     "zone_id": 4,
...     "zone_name": "Mystic Glade",
...     "zone_description": "Mystic Glade - Sanctuary of the Unicorns"
...   },
...   {
...     "zone_id": 5,
...     "zone_name": "Rainbow Valley",
...     "zone_description": "Rainbow Valley - Vibrant Refuge of the Unicorns"
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6477a9345500874ee8159166"),
    '1': ObjectId("6477a9345500874ee8159167"),
    '2': ObjectId("6477a9345500874ee8159168"),
    '3': ObjectId("6477a9345500874ee8159169"),
    '4': ObjectId("6477a9345500874ee815916a")
  }
}
```

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

3. Проверьте содержание коллекции единорогов.

```
learn> db.unicorns.updateOne(
...   { name: 'Ayna' },
...   { $set: { habitat: { $ref: 'unicorn_habitats', $id: 2 } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>

learn> db.unicorns.updateOne(
...   { name: 'Horny' },
...   { $set: { habitat: { $ref: 'unicorn_habitats', $id: 3 } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>

learn> db.unicorns.updateOne(
...   { name: 'Dunx' },
...   { $set: { habitat: { $ref: 'unicorn_habitats', $id: 4 } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>

learn> db.unicorns.updateOne(
...   { name: 'Kenny' },
...   { $set: { habitat: { $ref: 'unicorn_habitats', $id: 5 } } }
... )
{
  acknowledged: true,
  insertedId: null,
```

4. Содержание коллекции единорогов unicorns:

```
db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight:
600, gender: 'm', vampires: 63});

db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight:
450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'],
weight: 984, gender: 'm', vampires: 182});

db.unicorns.insert({name: 'Roooooodles', 44), 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'});

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

Задание 8.3.2:

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

```
learn> db.unicorns.createIndex({name: 1}, {unique: 1})
MongoServerError: Index build failed: 53a305fc-57c7-4a5f-a0c2-1ac4449a8fbc: Collection learn.unicorns ( e32ddd87-7269-406b-92c3-d3882c02af8e ) :: caused by :: E11000 duplicate
 key error collection: learn.unicorns index: name_1 dup key: { name: "Nimue" }
learn> db.unicorns.deleteOne({name: 'Nimue'})
{ acknowledged: true, deletedCount: 1 }
learn> db.unicorns.createIndex({name: 1}, {unique: 1})
MongoServerError: Index build failed: bb61c731-0e6a-4549-9890-38ea0de497d4: Collection learn.unicorns ( e32ddd87-7269-406b-92c3-d3882c02af8e ) :: caused by :: E11000 duplicate
 key error collection: learn.unicorns index: name_1 dup key: { name: "Pilot" }
learn> db.unicorns.deleteOne({name: 'Pilot'})
{ acknowledged: true, deletedCount: 1 }
learn> db.unicorns.createIndex({name: 1}, {unique: 1})
name_1
```

Как можно заметить, были дубликаты. После их удаления появилась возможность создать индекс для ключа *name*.

2. Содержание коллекции единорогов unicorns:

```
db.unicorns.insert({name: 'Horny', dob: new Date(1992,2,13,7,47), loves:
['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});

db.unicorns.insert({name: 'Aurora', dob: new Date(1991, 0, 24, 13, 0),
loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', dob: new Date(1973, 1, 9, 22, 10),
loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires:
182});

db.unicorns.insert({name: 'Roooooodles', dob: new Date(1979, 7, 18, 18,
44), loves: ['apple'], weight: 575, gender: 'm', vampires: 99});

db.unicorns.insert({name: 'Solnara', dob: new Date(1985, 6, 4, 2, 1),
loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f',
vampires:80});

db.unicorns.insert({name:'Ayna', dob: new Date(1998, 2, 7, 8, 30),
loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires:
40});

db.unicorns.insert({name:'Kenny', dob: new Date(1997, 6, 1, 10, 42),
loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});

db.unicorns.insert({name: 'Raleigh', dob: new Date(2005, 4, 3, 0, 57),
loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});

db.unicorns.insert({name: 'Leia', dob: new Date(2001, 9, 8, 14, 53),
loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires:
33});



db.unicorns.insert({name: 'Pilot', dob: new Date(1997, 2, 1, 5, 3),
loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires:
54});

db.unicorns.insert ({name: 'Nimue', dob: new Date(1999, 11, 20, 16, 15),
loves: ['grape', 'carrot'], weight: 540, gender: 'f'});

db.unicorns.insert {name: 'Dunx', dob: new Date(1976, 6, 18, 18, 18),
loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165
```

Задание 8.3.3:

1. Получите 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 }
]
```

2. Удалите все индексы, кроме индекса для идентификатора.

```
learn> db.unicorns.dropIndex('name_1')
{ nIndexesWas: 2, ok: 1 }
```

3. Попытайтесь удалить индекс для идентификатора.

```
learn> db.unicorns.dropIndex('_id_')
MongoServerError: cannot drop _id index
```

Задание 8.3.4:

Создайте 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("6477b67a5500874ee817180a") }
}
```

1) *Выберите последних четыре документа.*

```
learn> db.numbers.find().skip(99996)
[
  { _id: ObjectId("6477b67a5500874ee8171807"), value: 99996 },
  { _id: ObjectId("6477b67a5500874ee8171808"), value: 99997 },
  { _id: ObjectId("6477b67a5500874ee8171809"), value: 99998 },
  { _id: ObjectId("6477b67a5500874ee817180a"), value: 99999 }
]
```

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

```
learn> db.numbers.explain('executionStats').find().skip(99996)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'SKIP',
      skipAmount: 0,
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 49,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      stage: 'SKIP',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 100002,
      advanced: 4,
      needTime: 99997,
      needYield: 0,
      saveState: 100,
      restoreState: 100,
      isEOF: 1,
      skipAmount: 0,
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 100000,
        executionTimeMillisEstimate: 0,
        works: 100002,
        advanced: 100000,
        needTime: 1,
        needYield: 0,
        saveState: 100,
```

Понадобилось 49 мс.

3) *Создайте индекс для ключа* `value`.

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

4) *Получите информацию о всех индексах коллекции* `numbres`.

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

5) *Выполните запрос 2.*

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

```
learn> db.numbers.explain('executionStats').find().skip(99996)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'SKIP',
      skipAmount: 0,
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 42,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      stage: 'SKIP',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 100002,
      advanced: 4,
      needTime: 99997,
      needYield: 0,
      saveState: 100,
      restoreState: 100,
      isEOF: 1,
      skipAmount: 0,
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 100000,
        executionTimeMillisEstimate: 0,
        works: 100002,
        advanced: 100000,
        needTime: 1,
        needYield: 0,
        saveState: 100
```

В этот раз на выполнение запроса ушло 42 мс, что на 7 мс быстрее, чем без индекса.

7) *Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?*
Для маленького объема данных сложно оценить эффективность индексов. Иногда индексы могут “навредить” для такой маленькой коллекции.

Вывод: В ходе работы были рассмотрены основные аспекты организации и хранения данных в MongoDB, а также основные операции над данными, такие как вставка, редактирование, удаление и выборка с учетом условий. Было изучено использование агрегирующих функций, сортировки, ссылок для связи полей и значений разных документов, а также функций и курсоров для обработки результатов запросов.

Важным наблюдением было то, что использование индексирования ключей в коллекции не всегда приводит к значительному улучшению производительности простых запросов. Однако, для выполнения сложных запросов индексы могут быть полезными и позволить более эффективно выполнять запросы на больших объемах данных.

Также была изучена возможность просмотра статистики о произведенном запросе, что позволяет анализировать производительность и оптимизировать запросы для достижения более эффективных результатов.

В целом, работа с MongoDB предоставляет гибкую и мощную среду для работы с данными, позволяя эффективно хранить, обрабатывать и извлекать информацию в соответствии с требованиями приложения.