Министерство науки и высшего образования Российской Федерации
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

О Т Ч Е Т

по лабораторной работе №6 Работа с БД в СУБД MongoDB

По дисциплине «Проектирование и реализация баз данных»

Автор: Турищев А. И.
Факультет: ИКТ
Группа: К3340
Преподаватель: Говорова М. М.

Санкт-Петербург,
2025

Содержание

1 Цель работы

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.

2 Практическое задание

Практическое задание:

1. Практическое задание 2.1.1
   (a) Создайте базу данных learn.
   (b) Заполните коллекцию единорогов unicorns
   (c) Используя второй способ, вставьте в коллекцию единорогов документ
   (d) Проверьте содержимое коллекции с помощью метода find.
2. Практическое задание 2.2.1
   (a) Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.
   (b) Найдите всех самок, которые любят carrot. Ограничьте этот список первой особью с помощью функций findOne и limit.
3. Практическое задание 2.2.2: Модифицируйте запрос для вывода списков самцов единорогов, исключив из результата информацию о предпотениях и поле.
4. Практическое задание 2.2.3: Вывести список единорогов в обратном порядке добавления.
5. Практическое задание 2.2.4: Вывести список единорогов с названием первого любимого предпочтения, исключив идентификатор.
6. Практическое задание 2.3.1: Вывести список самок единорогов весом от полутонны до 700 кг, исключив вывод идентификатора.
7. Практическое задание 2.3.2: Вывести список самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора.
8. Практическое задание 2.3.3: Найти всех единорогов, не имеющих ключ vampires.
9. Практическое задание 2.3.4: Вывести список упорядоченный список имен самцов единорогов с информацией об их первом предпочтении.
10. Практическое задание 3.1.1:
    (a) Создайте коллекцию towns, включающую следующие документы
    (b) Сформировать запрос, который возвращает список городов с независимыми мэрами (party="I"). Вывести только название города и информацию о мэре.
    (c) Сформировать запрос, который возвращает список беспартийных мэров (party отсутствует). Вывести только название города и информацию о мэре.
11. Практическое задание 3.1.2:

(a) Сформировать функцию для вывода списка самцов единорогов.

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

(c) Вывести результат, используя forEach.

12. Практическое задание 3.2.1: Вывести количество самок единорогов весом от полутонны до 600 кг.

13. Практическое задание 3.2.2: Вывести список предпочтений.

14. Практическое задание 3.2.3: Посчитать количество особей единорогов обоих полов.

15. Практическое задание 3.3.1:

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

    (b) Проверить содержимое коллекции unicorns.

16. Практическое задание 3.3.2:

    (a) Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.

    (b) Проверить содержимое коллекции unicorns.

17. Практическое задание 3.3.3: Для самца единорога Raleigh внести изменения в БД: теперь он любит рэдбул.

18. Практическое задание 3.3.4: Всем самцам единорогов увеличить количество убитых вапмиров на 5.

19. Практическое задание 3.3.5: Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.

20. Практическое задание 3.3.6: Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.

21. Практическое задание 3.3.7: Изменить информацию о самке единорога Aurora: теперь она любит еще и сахар, и лимоны.

22. Практическое задание 3.4.1:

    (a) Создайте коллекцию towns, включающую следующие документы:

    (b) Удалите документы с беспартийными мэрами.

    (c) Проверьте содержание коллекции.

    (d) Очистите коллекцию.

    (e) Просмотрите список доступных коллекций.

23. Практическое задание 4.1.1:

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

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

(c) Проверьте содержание коллекции единорогов

24. Практическое задание 4.2.1: Проверьте, можно ли задать для коллекции unicorns индекс для ключа name с флагом unique

25. Практическое задание 4.3.1:

    (a) Получите информацию о всех индексах коллекции unicorns

    (b) Удалите все индексы, кроме индекса для идентификатора.

    (c) Попытайтесь удалить индекс для идентификатора.

26. Практическое задание 4.4.1:

    (a) Создайте объемную коллекцию numbers, задействовав курсор: for(i = 0; i < 100000; i++)db.numbers.insert(value: i)

    (b) Выберите последних четыре документа.

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

    (d) Создайте индекс для ключа value.

    (e) Получите информацию о всех индексах коллекции numbres.

    (f) Выполните запрос 2.

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

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

3 Выполнение

3.1 Практическое задание 2.1.1

```
learn> db.unicorns.insertMany([
...    {name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender:
    'm', vampires: 63},
...    {name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender
   : 'f', vampires: 43},
...    {name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984,
   gender: 'm', vampires: 182},
...    {name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm
   ', vampires: 99},
...    {name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight
   :550, gender:'f', vampires:80},
...    {name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733,
   gender: 'f', vampires: 40},
...    {name:'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: '
   m', vampires: 39},
...    {name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender
   : 'm', vampires: 2},
...    {name: 'Leia', loves: ['apple', 'watermelon'], weight: 601,
   gender: 'f', vampires: 33},
```

```
...    {name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650,
   gender: 'm', vampires: 54},
...    {name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender:
    'f'}
... ]);
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68dca59ceb6c982b26ce5f5e'),
    '1': ObjectId('68dca59ceb6c982b26ce5f5f'),
    '2': ObjectId('68dca59ceb6c982b26ce5f60'),
    '3': ObjectId('68dca59ceb6c982b26ce5f61'),
    '4': ObjectId('68dca59ceb6c982b26ce5f62'),
    '5': ObjectId('68dca59ceb6c982b26ce5f63'),
    '6': ObjectId('68dca59ceb6c982b26ce5f64'),
    '7': ObjectId('68dca59ceb6c982b26ce5f65'),
    '8': ObjectId('68dca59ceb6c982b26ce5f66'),
    '9': ObjectId('68dca59ceb6c982b26ce5f67'),
    '10': ObjectId('68dca59ceb6c982b26ce5f68')
  }
}
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('68dca5dbeb6c982b26ce5f69') }
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f5e'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f5f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
```

```
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
  _id: ObjectId('68dca59ceb6c982b26ce5f60'),
  name: 'Unicrom',
  loves: [ 'energon', 'redbull' ],
  weight: 984,
  gender: 'm',
  vampires: 182
},
{
  _id: ObjectId('68dca59ceb6c982b26ce5f61'),
  name: 'Roooooodles',
  loves: [ 'apple' ],
  weight: 575,
  gender: 'm',
  vampires: 99
},
{
  _id: ObjectId('68dca59ceb6c982b26ce5f62'),
  name: 'Solnara',
  loves: [ 'apple', 'carrot', 'chocolate' ],
  weight: 550,
  gender: 'f',
  vampires: 80
},
{
  _id: ObjectId('68dca59ceb6c982b26ce5f63'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 733,
  gender: 'f',
  vampires: 40
},
{
  _id: ObjectId('68dca59ceb6c982b26ce5f64'),
  name: 'Kenny',
  loves: [ 'grape', 'lemon' ],
  weight: 690,
  gender: 'm',
  vampires: 39
},
{
  _id: ObjectId('68dca59ceb6c982b26ce5f65'),
  name: 'Raleigh',
  loves: [ 'apple', 'sugar' ],
  weight: 421,
  gender: 'm',
```

```
    vampires: 2
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f66'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f67'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f68'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('68dca5dbeb6c982b26ce5f69'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
```

3.2 Практическое задание 2.2.1

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f5f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f63'),
    name: 'Ayna',
```

```
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f66'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn> db.unicorns.find({gender: 'm'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('68dca5dbeb6c982b26ce5f69'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f5e'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f64'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'});
{
  _id: ObjectId('68dca59ceb6c982b26ce5f5f'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

```
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1);
[
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f5f'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

3.3 Практическое задание 2.2.2

```
    learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).
       limit(1)
[
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f5e'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
]
```

3.4 Практическое задание 2.2.3

```
learn> db.unicorns.find().sort({$natural: -1}).limit(3)
[
  {
    _id: ObjectId('68dca5dbeb6c982b26ce5f69'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f68'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f67'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
```

```
    gender: 'm',
    vampires: 54
  }
]
```

3.5 Практическое задание 2.2.4

```
learn> db.unicorns.find({}, {name: 1, loves: {$slice: 1}, _id: 0})
    .limit(2)
[
  { name: 'Horny', loves: [ 'carrot' ] },
  { name: 'Aurora', loves: [ 'carrot' ] }
]
```

3.6 Практическое задание 2.3.1

```
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'
  }
]
```

3.7 Практическое задание 2.3.2

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

```
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

3.8 Практическое задание 2.3.3

```
    learn> db.unicorns.find({vampires: {$exists: 0}})
[
  {
    _id: ObjectId('68dca59ceb6c982b26ce5f68'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

3.9 Практическое задание 2.3.4

```
    learn> db.unicorns.find({gender: 'm'}, {name: 1, loves: {$slice:
       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: 'Roooooodles', loves: [ 'apple' ] },
  { name: 'Unicrom', loves: [ 'energon' ] }
]
```

3.10 Практическое задание 3.1.1

```
    learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     populatiuon: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: [""],
...     mayor: { name: "Jim Wehrle" }
...   },
...   {
...     name: "New York",
...     populatiuon: 22200000,
...     last_sensus: ISODate("2009-07-31"),
```

```
...       famous_for: ["status of liberty", "food"],
...       mayor: { name: "Michael Bloomberg", party: "I" }
...     },
...     {
...       name: "Portland",
...       populatiuon: 528000,
...       last_sensus: ISODate("2009-07-20"),
...       famous_for: ["beer", "food"],
...       mayor: { name: "Sam Adams", party: "D" }
...     }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68dcaf1eeb6c982b26ce5f6d'),
    '1': ObjectId('68dcaf1eeb6c982b26ce5f6e'),
    '2': ObjectId('68dcaf1eeb6c982b26ce5f6f')
  }
}
learn> db.towns.find({'mayor.party': 'I'}, {name: 1, mayor: 1, _id:
   0})
[
  {
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
learn> db.towns.find({'mayor.party': {$exists: 0}}, {name: 1, mayor:
   1, _id: 0})
[ { name: 'Punxsutawney', mayor: { name: 'Jim Wehrle' } } ]
```

3.11 Практическое задание 3.1.2

```
    learn> male = function() {return {gender: 'm'};}
[Function: male]
learn> var cursor = db.unicorns.find(male()); null;
null
learn> cursor.limit(2).sort({name: 1}); null
null
learn> cursor.forEach(function(obj) {print(obj);})
{
  _id: ObjectId('68dcb2e4eb6c982b26ce5f7b'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('68dcb2e4eb6c982b26ce5f70'),
```

```
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

3.12 Практическое задание 3.2.1

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

3.13 Практическое задание 3.2.2

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

3.14 Практическое задание 3.2.3

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

3.15 Практическое задание 3.3.1

save - deprecated в моей версии, добавляю и проверяю через insertOne()

```
learn> db.unicorns.insertOne({name: 'Barny', loves: ['grape'],
   weight: 340, gender: 'm'});
{
  acknowledged: true,
  insertedId: ObjectId('68dcb831eb6c982b26ce5f7c')
}
learn> db.unicorns.find({name: 'Barny'})
[
  {
    _id: ObjectId('68dcb831eb6c982b26ce5f7c'),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
```

```
  }
]
```

3.16 Практическое задание 3.3.2

Не давал обновить без atomic operators

```
    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('68dcb2e4eb6c982b26ce5f75'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

3.17 Практическое задание 3.3.3

```
    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('68dcb2e4eb6c982b26ce5f77'),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

3.18 Практическое задание 3.3.4

```
    learn> db.unicorns.find({gender: 'm'}).limit(2);
[
  {
    _id: ObjectId('68dcb2e4eb6c982b26ce5f70'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68dcb2e4eb6c982b26ce5f72'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
learn> db.unicorns.update({gender: 'm'}, {$inc: {vampires: 5}}, {multi
   : 1})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
learn> db.unicorns.find({gender: 'm'}).limit(2);
[
  {
    _id: ObjectId('68dcb2e4eb6c982b26ce5f70'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('68dcb2e4eb6c982b26ce5f72'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  }
]
```

3.19 Практическое задание 3.3.5

```
    learn> db.towns.update({name: 'Portland'}, {$unset: {'mayor.party
       ': 1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find({name: 'Portland'})
[
  {
    _id: ObjectId('68dcaf1eeb6c982b26ce5f6f'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

3.20 Практическое задание 3.3.6

```
    learn> db.unicorns.update({name: 'Pilot'}, {$push: {loves: '
       chocolate'}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne,
   updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Pilot'})
[
  {
    _id: ObjectId('68dcb2e4eb6c982b26ce5f79'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

3.21 Практическое задание 3.3.7

```
    learn> db.unicorns.update({name: 'Aurora'}, {$addToSet: {loves: {
       $each: ['lemon', 'sugar']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Aurora'})
[
  {
    _id: ObjectId('68dcb2e4eb6c982b26ce5f71'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'lemon', 'sugar' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

3.22 Практическое задание 3.4.1

```
    learn> db.towns.deleteMany({'mayor.party': {$exists: 0}})
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId('68dcc0efa2c0319050ce5f48'),
    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('68dcc0efa2c0319050ce5f49'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 2 }
learn> db.getCollectionNames()
[ 'unicorns', 'towns' ]
```

3.23 Практическое задание 4.1.1

```
    learn> db.habitats.insertMany[ {_id: 'forest', name: 'Magic Forest
       ', description: 'Magical woodland'}, {_id: 'mountain', name: '
       mountain Everest', description: 'Highest mountain'}]
learn> db.unicorns.update({gender: 'm'}, {$set: {habitat: {$ref: '
   habitats', $id: 'forest'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.update({gender: 'm'}, {$set: {habitat: {$ref: '
   habitats', $id: 'forest'}}}, {multi: 1})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 7,
  upsertedCount: 0
}
learn> db.unicorns.update({gender: 'f'}, {$set: {habitat: {$ref: '
   habitats', $id: 'mountain'}}}, {multi: 1})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 5,
  modifiedCount: 5,
  upsertedCount: 0
}
learn> db.unicorns.find({gender: 'm'}).limit(2);
[
  {
    _id: ObjectId('68dcc7c6a2c0319050ce5f4a'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63,
    habitat: DBRef('habitats', 'forest')
  },
  {
    _id: ObjectId('68dcc7c6a2c0319050ce5f4c'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182,
```

```
    habitat: DBRef('habitats', 'forest')
  }
]
learn> db.unicorns.find({gender: 'f'}).limit(2);
[
  {
    _id: ObjectId('68dcc7c6a2c0319050ce5f4b'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef('habitats', 'mountain')
  },
  {
    _id: ObjectId('68dcc7c6a2c0319050ce5f4e'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80,
    habitat: DBRef('habitats', 'mountain')
  }
]
```

3.24 Практическое задание 4.2.1

Можно

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

3.25 Практическое задание 4.3.1

```
    learn> db.unicorns.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
learn> db.unicorns.dropIndex("name_1")
{ nIndexesWas: 2, ok: 1 }
learn> db.unicorns.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn> db.unicorns.dropIndex("_id_")
MongoServerError[InvalidOptions]: cannot drop _id index
```

3.26 Практическое задание 4.4.1

```
learn> db.numbers.find().count()
100000
learn> db.numbers.find({value: {$gte: 99996}})
[
  { _id: ObjectId('68dcccaaa2c0319050d03e84'), value: 99996 },
  { _id: ObjectId('68dcccaaa2c0319050d03e85'), value: 99997 },
  { _id: ObjectId('68dcccaaa2c0319050d03e86'), value: 99998 },
  { _id: ObjectId('68dcccaaa2c0319050d03e87'), value: 99999 }
]
learn> db.numbers.explain("executionStats").find({value: {$gte:
   99996}})
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: { value: { '$gte': 99996 } },
    indexFilterSet: false,
    queryHash: 'AA258D80',
    planCacheShapeHash: 'AA258D80',
    planCacheKey: 'C5B6FEE0',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { value: { '$gte': 99996 } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 34,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { value: { '$gte': 99996 } },
      nReturned: 4,
      executionTimeMillisEstimate: 28,
      works: 100001,
      advanced: 4,
      needTime: 99996,
      needYield: 0,
      saveState: 1,
```

```
      restoreState: 1,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 100000
    }
  },
  queryShapeHash: '
     AF8E09276B8C4FF9FBBAA4ACB7DA4D92C8E99DADA95E5977ED7C6FBA55D93BBA
     ',
  command: {
    find: 'numbers',
    filter: { value: { '$gte': 99996 } },
    '$db': 'learn'
  },
  serverInfo: {
    host: '8b994f4fd3f6',
    port: 27017,
    version: '8.0.14',
    gitVersion: 'bbdb887c2ac94424af0ee8fcaad39203bdf98671'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
  },
  ok: 1
}
learn> db.numbers.ensureIndex({value: 1})
[ 'value_1' ]
learn> db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
learn> db.numbers.explain("executionStats").find({value: {$gte:
   99996}})
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: { value: { '$gte': 99996 } },
    indexFilterSet: false,
    queryHash: 'AA258D80',
```

```
    planCacheShapeHash: 'AA258D80',
    planCacheKey: 'C38141D5',
    optimizationTimeMillis: 1,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { value: 1 },
        indexName: 'value_1',
        isMultiKey: false,
        multiKeyPaths: { value: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { value: [ '[99996, inf.0]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 1,
    totalKeysExamined: 4,
    totalDocsExamined: 4,
    executionStages: {
      isCached: false,
      stage: 'FETCH',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      docsExamined: 4,
      alreadyHasObj: 0,
      inputStage: {
        stage: 'IXSCAN',
        nReturned: 4,
        executionTimeMillisEstimate: 0,
```

```
      works: 5,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      keyPattern: { value: 1 },
      indexName: 'value_1',
      isMultiKey: false,
      multiKeyPaths: { value: [] },
      isUnique: false,
      isSparse: false,
      isPartial: false,
      indexVersion: 2,
      direction: 'forward',
      indexBounds: { value: [ '[99996, inf.0]' ] },
      keysExamined: 4,
      seeks: 1,
      dupsTested: 0,
      dupsDropped: 0
    }
  }
},
queryShapeHash: '
   AF8E09276B8C4FF9FBBAA4ACB7DA4D92C8E99DADA95E5977ED7C6FBA55D93BBA
   ',
command: {
  find: 'numbers',
  filter: { value: { '$gte': 99996 } },
  '$db': 'learn'
},
serverInfo: {
  host: '8b994f4fd3f6',
  port: 27017,
  version: '8.0.14',
  gitVersion: 'bbdb887c2ac94424af0ee8fcaad39203bdf98671'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted',
  internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
```

```
  ok: 1
}
```

В моём случае индекс увеличил скорость в 34 раза, с 34 миллисекунд до 1 миллисекунды. Благодаря индексу, вместо того, чтобы анализировать все 100 000 документов, обрабатывались только 4 нужных.

4 Вывод

В ходе работы освоены основные операции MongoDB: вставка, выборка, обновление и удаление данных. Изучены методы фильтрации с использованием логических операторов, работа с вложенными документами и массивами, агрегация данных, управление индексами и создание ссылок между коллекциями. Практически подтверждено, что индексы повышают эффективность поиска. Получены навыки построения и оптимизации запросов в данной СУБД.