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

Лабораторная работа № 6.

Тема работы: «Работа с БД в СУБД MongoDB»

Обучающийся: Майстренко Анастасия Николаевна, K3241

Преподаватель: Говорова М.М.

Санкт-Петербург,
2025

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.

Программное обеспечение

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

Выполнение работы

Практическое задание 2.1.1:

1) Создайте базу данных *learn*.
2) Заполните коллекцию единорогов *unicorns*.

```
test> use learn
switched to db learn
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: { '0': ObjectId('684332a96b141b5c957e3ce8') }
}
learn>
```

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.insertOne(document)
...
{
  acknowledged: true,
  insertedId: ObjectId('684333c76b141b5c957e3ce9')
}
learn>
```

4) Проверьте содержимое коллекции с помощью метода *find*.

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('684332a96b141b5c957e3cde'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('684332a96b141b5c957e3cdf'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce0'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce1'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce2'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
```

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.

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('684332a96b141b5c957e3cdf'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce3'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce6'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn>
```

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('684333c76b141b5c957e3ce9'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('684332a96b141b5c957e3cde'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce4'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce7'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce5'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce1'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce0'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
learn>
```

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('684332a96b141b5c957e3cdf'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```

```
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId('684332a96b141b5c957e3cdf'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

Практическое задание 2.2.2:

1) 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('684332a96b141b5c957e3cde'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce0'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce1'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce4'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce5'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce7'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('684333c76b141b5c957e3ce9'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
]
learn>
```

Практическое задание 2.2.3:

1) Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({$natural: -1})
...
[
  {
    _id: ObjectId('684333c76b141b5c957e3ce9'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce8'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce7'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce6'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce5'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce4'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce3'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
```

Практическое задание 2.2.4:

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

```
learn> db.unicorns.find({}, {name: 1, loves: {$slice: 1}, _id: 0})
[...
[
  { name: 'Horny', loves: [ 'carrot' ] },
  { name: 'Aurora', loves: [ 'carrot' ] },
  { name: 'Unicrom', loves: [ 'energon' ] },
  { name: 'Roooooodles', loves: [ 'apple' ] },
  { name: 'Solnara', loves: [ 'apple' ] },
  { name: 'Ayna', loves: [ 'strawberry' ] },
  { name: 'Kenny', loves: [ 'grape' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Leia', loves: [ 'apple' ] },
  { name: 'Pilot', loves: [ 'apple' ] },
  { name: 'Nimue', loves: [ 'grape' ] },
  { name: 'Dunx', loves: [ 'grape' ] }
]
learn>
```

Практическое задание 2.3.1:

1) Вывести список самок единорогов весом от 500 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'
  }
]
learn>
```

Практическое задание 2.3.2:

1) Вывести список самцов единорогов весом от 500кг 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:

1) Найти всех единорогов, не имеющих ключ *vampires*.

```
learn> db.unicorns.find({vampires: {$exists: false}})
[...
[
  {
    _id: ObjectId('684332a96b141b5c957e3ce8'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn>
```

Практическое задание 2.3.4:

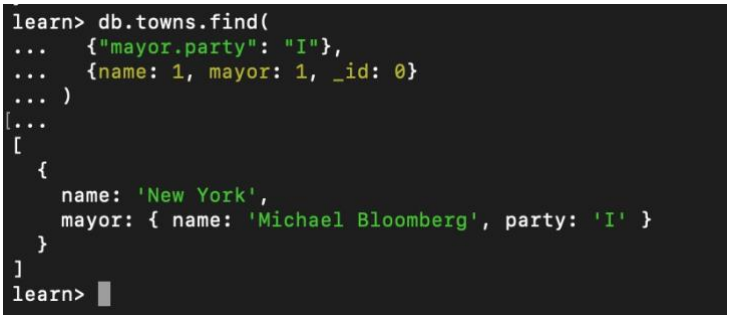
1) Вывести список упорядоченный имен самцов единорогов с информацией об их первом предпочтении.

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

Практическое задание 3.1.1:

1) Создайте коллекцию *towns*, включающую следующие документы:

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



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

Практическое задание 3.1.2:

1) Сформировать функцию для вывода списка самцов единорогов.
2) Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.
3) Вывести результат, используя *forEach*.
4) Содержание коллекции единорогов *unicorns*:

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

learn>
```

Практическое задание 3.2.1:

1) Вывести количество самок единорогов весом от 500 до 600 кг.

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

Практическое задание 3.2.2:

1) Вывести список предпочтений.

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

Практическое задание 3.2.3:

1) Посчитать количество особей единорогов обоих полов.

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

Практическое задание 3.3.1:

1) Выполнить команду:

``` *db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})* ```2) Проверить содержимое коллекции *unicorns*.
```

```
learn> db.unicorns.insertOne({
...   name: 'Barny',
...   loves: ['grape'],
...   weight: 340,
...   gender: 'm'
... })
[...
{
  acknowledged: true,
  insertedId: ObjectId('68433ebd6b141b5c957e3ced')
}
learn> db.unicorns.find({name: 'Barny'})
[...
[
  {
    _id: ObjectId('68433ebd6b141b5c957e3ced'),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
learn>
```

Практическое задание 3.3.2:

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

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

2) Проверить содержимое коллекции *unicorns*.

```
learn> db.unicorns.find({name: "Ayna"})
[...
[
  {
    _id: ObjectId('684332a96b141b5c957e3ce3'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
learn>
```

Практическое задание 3.3.3:

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

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

2) Проверить содержимое коллекции *unicorns*.

```
learn> db.unicorns.find({name: "Raleigh"})
[...
[
  {
    _id: ObjectId('684332a96b141b5c957e3ce5'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 7
  }
]
learn>
```

Практическое задание 3.3.4:

1) Всем самцам единорогов увеличить количество убитых вапмиров на 5.

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

2) Проверить содержимое коллекции *unicorns*.

```
learn> db.unicorns.find({gender: "m"}).limit(3)
[...
[
  {
    _id: ObjectId('684332a96b141b5c957e3cde'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 73
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce0'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 192
  },
  {
    _id: ObjectId('684332a96b141b5c957e3ce1'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 109
  }
]
learn>
```

Практическое задание 3.3.5:

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

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

2) Проверить содержимое коллекции *towns*.

```
learn> db.towns.find({name: "Portland"})
[
  {
    _id: ObjectId('68433b5c6b141b5c957e3cec'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
learn>
```

Практическое задание 3.3.6:

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

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

2) Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.find({name: "Pilot"})
[
  {
    _id: ObjectId('684332a96b141b5c957e3ce7'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 64
  }
]
learn>
```

Практическое задание 3.3.7:

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

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

2. *Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.find({name: "Aurora"})
[
  {
    _id: ObjectId('684332a96b141b5c957e3cdf'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

Практическое задание 3.4.1:

1) Создайте коллекцию *towns*, включающую следующие документы:

```
learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     populatiuon: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: ["phil the groundhog"],
...     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('6843434d6b141b5c957e3cee'),
    '1': ObjectId('6843434d6b141b5c957e3cef'),
    '2': ObjectId('6843434d6b141b5c957e3cf0')
  }
}
learn>
```

2) Удалите документы с беспартийными мэрами.

```
learn> db.towns.deleteMany({"mayor.party": {$exists: false}})
...
{ acknowledged: true, deletedCount: 0 }
```

3) Проверьте содержание коллекции.

```
learn> db.towns.find()
[
  {
    _id: ObjectId('6843434d6b141b5c957e3cef'),
    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('6843434d6b141b5c957e3cf0'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>
```

4) Очистите коллекцию.

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

learn>
```

5) Просмотрите список доступных коллекций.

```
learn> show collections
[...
towns
unicorns
```

Практическое задание 4.1.1:

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

```
learn> db.habitats.insertMany([
...   {
...     _id: "forest",
...     full_name: "Enchanted Forest",
...     description: "A magical place full of ancient trees and glowing mushrooms."
...   },
...   {
...     _id: "mountains",
...     full_name: "Crystal Mountains",
...     description: "High peaks where the air sparkles with magic dust."
...   },
...   {
...     _id: "desert",
...     full_name: "Golden Dunes",
...     description: "Hot and mysterious sands where few dare to travel."
...   }
... ])
[...
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'mountains', '2': 'desert' }
}
learn>
```

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

```
[learn> db.unicorns.update({name: "Dunx"}, {$set: {habitat: {$ref: "habitats", $id: "forest"}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
[learn> db.unicorns.update({name: "Pilot"}, {$set: {habitat: {$ref: "habitats", $id: "desert"}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>
```

3) Проверьте содержание коллекции едиорогов.

```
learn> db.unicorns.find({name: "Dunx"})
[
  {
    _id: ObjectId('684333c76b141b5c957e3ce9'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 175,
    habitat: DBRef('habitats', 'forest')
  }
]
learn>
```

```
learn> db.unicorns.find({name: "Pilot"})
[
  {
    _id: ObjectId('684332a96b141b5c957e3ce7'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 64,
    habitat: DBRef('habitats', 'desert')
  }
]
learn>
```

Практическое задание 4.2.1:

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

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

Практическое задание 4.3.1:

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 }
]
learn>
```

2) Удалите все индексы, кроме индекса для идентификатора.

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

3) Попытайтесь удалить индекс для идентификатора.

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

Практическое задание 4.4.1:

1) Создайте объемную коллекцию numbers, задействовав курсор: ```*for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}*```

```
learn> let cursor = Array.from({length: 100000}, (_, i) => ({value: i})).values();
... while (true) {
...     let next = cursor.next();
...     if (next.done) break;
...     db.numbers.insertOne(next.value);
... }
[...

{
  acknowledged: true,
  insertedId: ObjectId('68434e2b6b141b5c957fc390')
}
learn>
```

2) Выберите последних четыре документа.

```
learn> db.numbers.find().sort({$natural: -1}).limit(4)
[...
[
  { _id: ObjectId('68434e2b6b141b5c957fc390'), value: 99999 },
  { _id: ObjectId('68434e2b6b141b5c957fc38f'), value: 99998 },
  { _id: ObjectId('68434e2b6b141b5c957fc38e'), value: 99997 },
  { _id: ObjectId('68434e2b6b141b5c957fc38d'), value: 99996 }
]
learn>
```

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

```
  nReturned: 1,
  executionTimeMillis: 2,
```

4) Создайте индекс для ключа value.

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

5) 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' }
]
learn>
```

6) Выполните запрос 2.

```
learn> db.numbers.find().sort({$natural: -1}).limit(4)
[...
[
  { _id: ObjectId('68434e2b6b141b5c957fc390'), value: 99999 },
  { _id: ObjectId('68434e2b6b141b5c957fc38f'), value: 99998 },
  { _id: ObjectId('68434e2b6b141b5c957fc38e'), value: 99997 },
  { _id: ObjectId('68434e2b6b141b5c957fc38d'), value: 99996 }
]
learn>
```

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

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

Ответ: Запрос с использованием индекса выполняется значительно быстрее (около 0 мс) по сравнению с запросом без индекса (2 мс), так как индекс позволяет избежать полного просмотра коллекции. Запрос с использованием индекса намного эффективнее.

Контрольные вопросы:

Пункт 3

1) **Как используется оператор точка?**

   Оператор точка (.) используется для доступа к полям вложенных документов и элементам массивов.

2) **Как можно использовать курсор?**

   Курсор позволяет итерироваться по результатам запроса. Его можно использовать для:
   - Постепенной обработки больших наборов данных.
   - Применения методов, таких как *.limit()*, *.skip()*, *.sort()*.
   - Преобразования результатов с помощью *.map()* или *.forEach()*.

3) **Какие возможности агрегирования данных существуют в MongoDB?**

   MongoDB предоставляет мощный Aggregation Framework, включающий:
   - Этапы пайплайна (*$match*, *$group*, *$sort*, *$project*, *$lookup* и др.).
   - Операторы агрегации (*$sum, $avg*, *$max*, *$min*, *$push* и др.).
   - Вычисления и преобразования данных (арифметика, строковые операции, работа с датами).
   - Джойны между коллекциями (*$lookup*).

4) **Какая из функций *save* или *update* более детально позволит настроить редактирование документов коллекции?**

   Метод *update* (или *updateOne/updateMany*) предоставляет более гибкие настройки, так как позволяет:
   - Точечно изменять поля с помощью операторов (*$set*, *$unset*, *$inc*).
   - Использовать условия для выбора документов.
   - Применять сложные модификации с агрегационными операциями (*$addToSet*, *$pull*).
   - Метод *save* просто перезаписывает документ (если *_id* существует) или вставляет новый (если *_id* нет).

5) **Как происходит удаление документов из коллекции по умолчанию?.**

   По умолчанию методы *deleteOne()* и *deleteMany()* физически удаляют документы из коллекции.

Пункт 4

1) Назовите способы связывания коллекций в MongoDB.

- Вложение документов (Embedding) – хранение связанных данных внутри одного документа.
- Ссылки (Reference) – хранение *ObjectId* и использование *$lookup* для джойнов.
- Денормализация – дублирование данных для ускорения чтения.

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

В MongoDB нет жесткого ограничения на количество индексов, но рекомендуется не более 64 индексов на коллекцтию.

3) Как получить информацию о всех индексах базы данных MongoDB?

db.collection.getIndexes()

Вывод:

В процессе выполнения лабораторной работы я освоила основные принципы работы с MongoDB. Были изучены базовые операции по управлению данными: создание, чтение, обновление и удаление документов (CRUD-операции). Особое внимание уделялось работе со сложными структурами данных, включая вложенные документы и массивы.

Полученные знания дают мне понимание того, как эффективно использовать MongoDB в реальных проектах. Я научилась выбирать оптимальные подходы к хранению и обработке данных в зависимости от конкретных задач. Эти навыки будут полезны при разработке современных приложений, работающих с большими объемами информации.