Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«Работа с БД в СУБД MongoDB»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Машковцева Марина Алексеевна
Факультет прикладной информатики
Группа K3240
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ 3
2. ПРАКТИЧЕСКОЕ ЗАДАНИЕ 4
3. ВЫПОЛНЕНИЕ 5
4. ВЫВОДЫ 28

1. 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

1) CRUD-ОПЕРАЦИИ В СУБД MONGODB. ВСТАВКА ДАННЫХ. ВЫБОРКА ДАННЫХ

2) ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB. ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ

3) ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

3. ВЫПОЛНЕНИЕ

Практическое задание 2.1.1:

1) Создайте базу данных learn.

2) Заполните коллекцию 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', 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'});

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

{name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}

4) Проверьте содержимое коллекции с помощью метода find.

Выполнение:

1)

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

2)

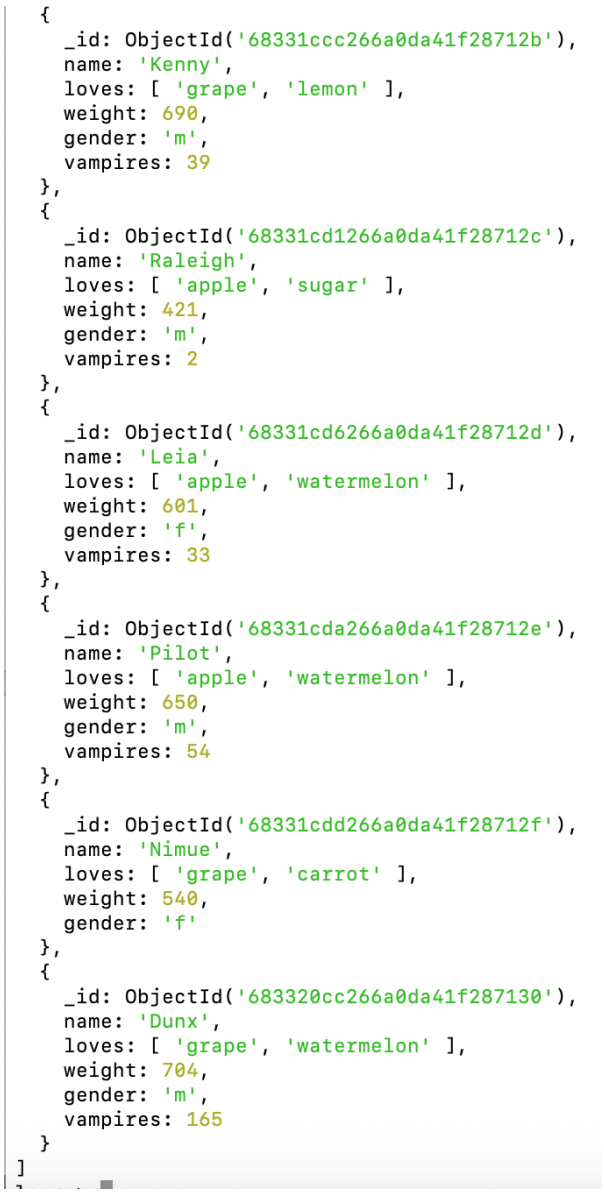
```
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 4
50, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68331cb1266a0da41f287126') }
}
```

3)

```
learn> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gen
der: 'm', vampires: 165}
[...                                                                            ]
[...                                                                            ]
[... )                                                                         ]
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683320cc266a0da41f287130') }
}
```

4)

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('68331ca8266a0da41f287125'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68331cb1266a0da41f287126'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('68331cb6266a0da41f287127'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('68331cbd266a0da41f287128'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('68331cc2266a0da41f287129'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('68331cc7266a0da41f28712a'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
```



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.

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

Выполнение:

1)

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1})
[
  {
    _id: ObjectId('683320cc266a0da41f287130'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('68331ca8266a0da41f287125'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68331ccc266a0da41f28712b'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('68331cda266a0da41f28712e'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('68331cd1266a0da41f28712c'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('68331cbd266a0da41f287128'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('68331cb6266a0da41f287127'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
```

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('68331cb1266a0da41f287126'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('68331cc7266a0da41f28712a'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('68331cd6266a0da41f28712d'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

2)
Вариант с findOne (вернёт одну случайную запись):

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId('68331cb1266a0da41f287126'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Вариант с limit (вернёт первую запись из упорядоченного списка):

```
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId('68331cb1266a0da41f287126'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 2.2.2:

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

Выполнение:

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
[
  {
    _id: ObjectId('68331ca8266a0da41f287125'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('68331cb6266a0da41f287127'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId('68331cbd266a0da41f287128'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('68331ccc266a0da41f28712b'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('68331cd1266a0da41f28712c'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('68331cda266a0da41f28712e'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('683320cc266a0da41f287130'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
]
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

Выполнение:

```
learn> db.unicorns.find().sort({ $natural: -1 })
 [
   {
     _id: ObjectId('683320cc266a0da41f287130'),
     name: 'Dunx',
     loves: [ 'grape', 'watermelon' ],
     weight: 704,
     gender: 'm',
     vampires: 165
   },
   {
     _id: ObjectId('68331cdd266a0da41f28712f'),
     name: 'Nimue',
     loves: [ 'grape', 'carrot' ],
     weight: 540,
     gender: 'f'
   },
   {
     _id: ObjectId('68331cda266a0da41f28712e'),
     name: 'Pilot',
     loves: [ 'apple', 'watermelon' ],
     weight: 650,
     gender: 'm',
     vampires: 54
   },
   {
     _id: ObjectId('68331cd6266a0da41f28712d'),
     name: 'Leia',
     loves: [ 'apple', 'watermelon' ],
     weight: 601,
     gender: 'f',
     vampires: 33
   },
   {
     _id: ObjectId('68331cd1266a0da41f28712c'),
     name: 'Raleigh',
     loves: [ 'apple', 'sugar' ],
     weight: 421,
     gender: 'm',
     vampires: 2
   },
```

Практическое задание 2.1.4:

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

Выполнение:

```
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:

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

Выполнение:

```
learn> db.unicorns.find({weight: {$gte: 500, $lte: 700}}, {_id: 0})
[
  {
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
```

Практическое задание 2.3.2:

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

Выполнение:

```
learn> db.unicorns.find({gender: 'm', weight: {$gte: 500}, loves: {$in: ['grape', 'lemon']}}
,{_id: 0})
[
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
```

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

Выполнение:

```
learn> db.unicorns.find({vampires: {$exists:false}})
[
  {
    _id: ObjectId('68331cdd266a0da41f28712f'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.4:

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

Выполнение:

```
learn> db.unicorns.find({gender: 'm'}, {loves: { $slice: 1}, _id: 0, weight: 0, vampires: 0,
 gender: 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' ] }
]
```

Практическое задание 3.1.1:

1) Создайте коллекцию towns, включающую следующие документы:

{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"}}

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 беспартийных мэров (party отсутствует). Вывести только название города и информацию о мэре.

Выполнение:

1)

```
test> use towns
switched to db towns
towns> db.towns.insert({
... name: "Punxsutawney",
...   population: 6200,
...   last_sensus: new ISODate("2008-01-31"),
...   famous_for: [""],
...   mayor: {
...     name: "Jim Wehrle"
...   }
... })
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany
, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6834760c6be06fd73e4ae921') }
}
towns> db.towns.insert({
...   name: "New York",
...   population: 22200000,
...   last_sensus: new ISODate("2009-07-31"),
...   famous_for: ["statue of liberty", "food"],
...   mayor: {
...     name: "Michael Bloomberg",
...     party: "I"
...   }
... })
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683476176be06fd73e4ae922') }
}
towns> db.towns.insert({
...   name: "Portland",
...   population: 528000,
...   last_sensus: new ISODate("2009-07-20"),
...   famous_for: ["beer", "food"],
...   mayor: {
...     name: "Sam Adams",
```

2)

```
towns> db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 1})
[
  {
    _id: ObjectId('683476176be06fd73e4ae922'),
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
```

3)

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

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:

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})

Выполнение:

Если делать как в примере, то получаются ошибки:

```
unicorns> fn = function() {return this.gender=='m';}
 [Function: fn]
unicorns> db.unicorns.find(fn)
 MongoInvalidArgumentError: Query filter must be a plain object or ObjectId
unicorns> var cursor = db.unicorns.find(fn);null;
 null
unicorns> cursor.sort({name:1}).limit(2);null;
 null
unicorns> cursor.forEach(function (obj) {
... print(obj.name);
... })
 MongoInvalidArgumentError: Query filter must be a plain object or ObjectId
```

Ошибка MongoInvalidArgumentError: Query filter must be a plain object or ObjectId возникает, потому что MongoDB больше не поддерживает передачу JavaScript-функции как фильтра в find() начиная с более новых версий. Поэтому пришлось немного переделать синтаксис:

```
unicorns> var cursor = db.unicorns.find({ gender: 'm' }).sort({ name: 1 }).limit(2);

unicorns> cursor.forEach(function(unicorn) {
...     print(unicorn.name);
... });
Dunx
Horny
```

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

Выполнение:

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

Практическое задание 3.2.2:
Вывести список предпочтений.

Выполнение:

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

Практическое задание 3.2.3:
Посчитать количество особей единорогов обоих полов.

Выполнение:

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

Практическое задание 3.3.1:
1) Выполнить команду:
> db.unicorns.save({name: 'Barny', loves: ['grape'],
weight: 340, gender: 'm'})
2) Проверить содержимое коллекции unicorns.

Выполнение:

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

Аналог:

```
unicorns> db.unicorns.insertOne({
...   name: 'Barny',
...   loves: ['grape'],
...   weight: 340,
...   gender: 'm'
... })
{
  acknowledged: true,
  insertedId: ObjectId('683486f56be06fd73e4ae930')
}
```

Практическое задание 3.3.2:

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

2) Проверить содержимое коллекции unicorns.

Выполнение:

1)

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

2)

```
  _id: ObjectId('6834796e6be06fd73e4ae929'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 800,
  gender: 'f',
  vampires: 51
},
```

Практическое задание 3.3.3:

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

2) Проверить содержимое коллекции unicorns.

Выполнение:

1)

```
unicorns> db.unicorns.update({name: "Raleigh"}, {$set: {loves: "Redbull"}}, {multi:true}
)
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2)

```
{
  _id: ObjectId('683479796be06fd73e4ae92b'),
  name: 'Raleigh',
  loves: 'Redbull',
  weight: 421,
  gender: 'm',
  vampires: 2
},
```

Практическое задание 3.3.4:

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

2) Проверить содержимое коллекции unicorns.

Выполнение:

1)

```
unicorns> db.unicorns.update({gender: 'm'}, {$inc: {vampires:5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2)

```
unicorns> db.unicorns.find()
[
  {
    _id: ObjectId('683478fc6be06fd73e4ae924'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('683479016be06fd73e4ae925'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683479066be06fd73e4ae926'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683479616be06fd73e4ae927'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
```

Практическое задание 3.3.5:

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

2) Проверить содержимое коллекции towns.

Выполнение:

1)

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

2)

```
[
  {
    _id: ObjectId('683476206be06fd73e4ae923'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

Практическое задание 3.3.6:

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

2) Проверить содержимое коллекции unicorns.

Выполнение:

1)

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

2)

```
  _id: ObjectId('683479816be06fd73e4ae92d'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon', 'chocolate' ],
  weight: 650,
  gender: 'm',
  vampires: 54
},
```

Практическое задание 3.3.7:

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

2) Проверить содержимое коллекции unicorns.

Выполнение:

1)

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

2)

```
  _id: ObjectId('683479016be06fd73e4ae925'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
```

Практическое задание 3.4.1:

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"}}
```

2) Удалите документы с беспартийными мэрами.
3) Проверьте содержание коллекции.
4) Очистите коллекцию.
5) Просмотрите список доступных коллекций.

Выполнение:

1)

```
unicorns> db.towns.insertMany([
... {
...     name: "Punxsutawney",
...     population: 6200,
... last_sensus: ISODate("2008-01-31"),
...     famous_for: ["phil the groundhog"],
...     mayor: {
...       name: "Jim Wehrle"
...     }
...   },
...   {
...     name: "New York",
...     population: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["statue of liberty", "food"],
...     mayor: {
...       name: "Michael Bloomberg",
...       party: "I"
...     }
...   },
...   {
...     name: "Portland",
...     population: 528000,
...     last_sensus: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: {
...       name: "Sam Adams",
...       party: "D"
...     }
...   }
```

2, 3)

```
towns> db.towns.remove({"mayor.party": {$exists: false}})
{ acknowledged: true, deletedCount: 1 }
towns> db.towns.find()
[
  {
    _id: ObjectId('6834925b6be06fd73e4ae935'),
    name: 'New York',
    population: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('6834925b6be06fd73e4ae936'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

4)

```
towns> db.towns.remove({})
{ acknowledged: true, deletedCount: 2 }
```

5)

```
towns> show collections
towns
```

Практическое задание 4.1.1:

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

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

3) Проверьте содержание коллекции единорогов.

Выполнение:

1)

```
unicorns> db.habitats.insertMany([
...   {
...     _id: "forest",
...     fullName: "Enchanted Forest of Eldoria",
...     description: "Dense magical woodland where unicorns graze on silver-leafed plant
s and drink from crystal-clear springs."
...   },
...   {
...     _id: "mount",
...     fullName: "Mount Celestia Peaks",
...     description: "Snow-capped mountains where unicorns roam above the clouds, their
coats adapting to the icy winds."
...   },
...   {
...     _id: "meadow",
...     fullName: "Golden Prairie of the Dawn",
...     description: "Vast flower-filled plains that glow at sunrise, home to herds of p
layful unicorn foals."
...   }
... ])
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'mount', '2': 'meadow' }
}
```

2)

```
unicorns> db.unicorns.update({_id:ObjectId("683493de6be06fd73e4ae942")},{$set:{habitat:{
$ref:"habitats", $id: "forest"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
unicorns> db.unicorns.update({_id:ObjectId("683493ad6be06fd73e4ae941")},{$set:{habitat:{
$ref:"habitats", $id: "mount"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

3)

```
    _id: ObjectId('683493ad6be06fd73e4ae941'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f',
    habitat: DBRef('habitats', 'mount')
  },
  {
    _id: ObjectId('683493de6be06fd73e4ae942'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165,
    habitat: DBRef('habitats', 'forest')
  }
]
```

Практическое задание 4.2.1:

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

Выполнение:

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

Был создан индекс с именем name_1.

Практическое задание 4.3.1:

1) Получите информацию о всех индексах коллекции unicorns .
2) Удалите все индексы, кроме индекса для идентификатора.
3) Попытайтесь удалить индекс для идентификатора.

Выполнение:

1)

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

2)

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

3)

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

Практическое задание 4.4.1:

1) Создайте объемную коллекцию numbers, задействовав курсор:

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

2) Выберите последних четыре документа.

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

4) Создайте индекс для ключа value.

5) Получите информацию о всех индексах коллекции numbres.

6) Выполните запрос 2.

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?

Выполнение:

1)

```
unicorns> use numbers
switched to db numbers
numbers> for (let i = 0; i < 100000; i++) {
...     db.numbers.insertOne({ value: i });
... }
...
{
  acknowledged: true,
  insertedId: ObjectId('68349a826be06fd73e4c6fee')
}
```

2)

```
numbers> db.numbers.find().sort({ _id: -1 }).limit(4)
...
[
  { _id: ObjectId('68349a826be06fd73e4c6fee'), value: 99999 },
  { _id: ObjectId('68349a826be06fd73e4c6fed'), value: 99998 },
  { _id: ObjectId('68349a826be06fd73e4c6fec'), value: 99997 },
  { _id: ObjectId('68349a826be06fd73e4c6feb'), value: 99996 }
]
```

3)

```
numbers> db.numbers.explain("executionStats").find().sort({ _id: -1 }).limit(4)
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 0,
  totalKeysExamined: 4,
  totalDocsExamined: 4,
  executionStages: {
```

4)

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

5)

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

6)

```
numbers> db.numbers.find().sort({ _id: -1 }).limit(4)
[
  { _id: ObjectId('68349a826be06fd73e4c6fee'), value: 99999 },
  { _id: ObjectId('68349a826be06fd73e4c6fed'), value: 99998 },
  { _id: ObjectId('68349a826be06fd73e4c6fec'), value: 99997 },
  { _id: ObjectId('68349a826be06fd73e4c6feb'), value: 99996 }
]
```

7)

```
numbers> db.numbers.explain("executionStats").find().sort({ _id: -1 }).limit(4)
{
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 4,
    totalDocsExamined: 4,
    executionStages: {
```

8) В моем случае показатели выполнения получились одинаково маленькими, но в целом метод с индексами должен работать более эффективно, особенно при работе с фильтрацией, сортировкой и большими объёмами данных, потому что не тратится время на просмотр всех документов, а ищется точно и быстро по ключу.

4. ВЫВОДЫ

В ходе лабораторной работы были освоены основные операции работы с MongoDB, включая CRUD-операции, агрегацию данных, работу с индексами и ссылками. Успешно выполнены задания по созданию, чтению, обновлению и удалению документов, а также проведена оптимизация запросов с использованием индексов. Практика показала, что индексы значительно ускоряют выполнение запросов, особенно при работе с большими объемами данных. Работа с вложенными объектами и ссылками подтвердила гибкость и мощь MongoDB для управления сложными структурами данных. В результате были получены навыки, необходимые для эффективного проектирования и работы с базами данных в MongoDB.