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 Шафиков Максим Азатович
Факультет прикладной информатики
Группа K3239
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

Цель: овладеть практическими навыками работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.

Практическое задание и выполнение:

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

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

```
obj = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
db.unicorns.insertOne(obj)
{
  acknowledged: true,
  insertedId: ObjectId('682529fb84703266ba06bc3e')
}
```

Используя второй способ, вставьте в коллекцию единорогов документ:

```
> db.unicorns.find({gender: 'm'}).sort({name: 1})
< {
    _id: ObjectId('682529fb84703266ba06bc3e'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('682529d484703266ba06bc33'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
```

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
< {
    _id: ObjectId('682529d484703266ba06bc34'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('682529d484703266ba06bc38'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
```

Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
< {
    _id: ObjectId('682529d484703266ba06bc34'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
< {
    _id: ObjectId('682529d484703266ba06bc34'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f'
```

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

```
> db.unicorns.find({ gender: 'm' }, { loves: 0, gender: 0 })
< {
    _id: ObjectId('682529d484703266ba06bc33'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
  {
    _id: ObjectId('682529d484703266ba06bc35'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
  {
    _id: ObjectId('682529d484703266ba06bc36'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  }
  {
    _id: ObjectId('682529d484703266ba06bc39'),
```

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

```
> db.unicorns.find().sort({ $natural: -1 })
< {
    _id: ObjectId('682529fb84703266ba06bc3e'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('682529d484703266ba06bc3d'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

Вывести список единорогов в обратном порядке добавления.

```
> 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: [
```

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

```
> 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
```

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

```
> 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
  }
```

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

```
> db.unicorns.find({vampires: {$exists:false}})
< {
    _id: ObjectId('682529d484703266ba06bc3d'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

Найти всех единорогов, не имеющих ключ vampires.

```
> 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: [
```

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

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

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

```
> db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 1, _id: 0})
< {
    name: 'New York',
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
```

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

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

*Сформировать запрос, который возвращает список беспартийных мэров (*party *отсутствует*). *Вывести только название города и информацию о мэре.*

```
> var male = function() { return this.gender === 'm'; }
> var cursor = db.unicorns.find({ $where: male }).sort({ name: 1 }).limit(2);
> cursor.forEach(function(unicorn) {
  print(unicorn.name);
  })
< Dunx
< Horny
```

Сформировать функцию для вывода списка самцов единорогов. Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке. Вывести результат, используя forEach.

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

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Вывести список предпочтений.

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

Посчитать количество особей единорогов обоих полов.

```
> db.unicorns.update({name: "Ayna"}, { $set: {weight: 800, vampires: 51} })
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

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

```
> db.unicorns.updateOne({ name: "Raleigh" }, { $addToSet: { loves: "redbull" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

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

```
db.unicorns.updateMany({ gender: 'm' }, { $inc: { vampires: 5 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
```

Всем самцам единорогов увеличить количество убитых вапмиров на 5.

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

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

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

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

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

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

```
> db.towns.deleteMany({ "mayor.party": { $exists: false } })
< {
    acknowledged: true,
    deletedCount: 1
  }
```

Удалите документы с беспартийными мэрами.

```
> db.towns.deleteMany({})
< {
    acknowledged: true,
    deletedCount: 2
  }
> show collections
< laba
  learn
  towns
  unicorns
```

Очистите коллекцию. Просмотрите список доступных коллекций.

4. ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

```
> db.habitats.insertMany([
    {_id: "forest", name: "Forest", description: "Great Forest"},
    {_id: "river", name: "River", description: "Fast River"},
    {_id: "mountains", name: "Mountains", description: "High Mountains"},
    {_id: "city", name: "City", description: "Saint Petersburg"}
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': 'forest',
      '1': 'river',
      '2': 'mountains',
      '3': 'city'
    }
  }
```

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

```
> db.unicorns.updateOne(
    { name: "Horny" },
    { $set: { habitat: { $ref: "habitats", $id: "forest" } } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.updateOne(
    { name: "Aurora" },
    { $set: { habitat: { $ref: "habitats", $id: "river" } } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.updateOne(
```

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

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

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

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

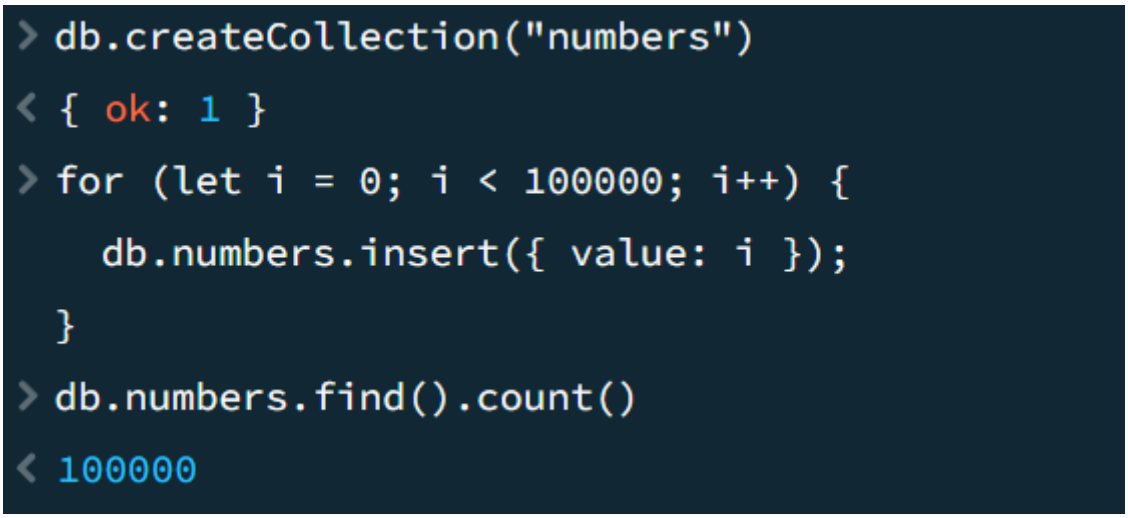
Получите информацию обо всех индексах коллекции unicorns. Удалите все индексы, кроме индекса для идентификатора. Попытайтесь удалить индекс для идентификатора.

Задание 4.4

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?*



Вставил 100000 записей

Выберите последних четыре документа. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра executionTimeMillis)

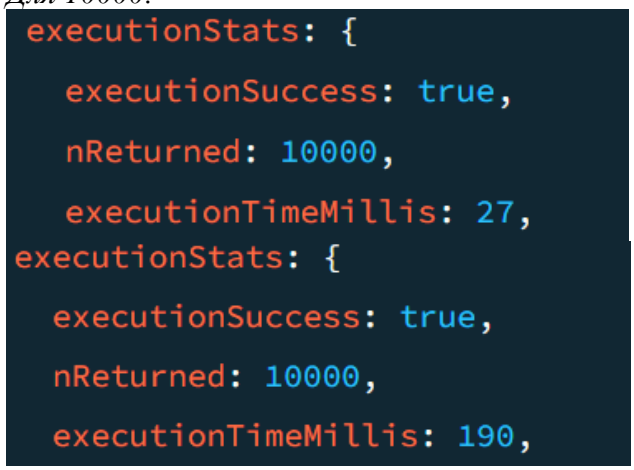
Вывело, что времени понадобилось 11 мс

Создайте индекс для ключа value. Получите информацию обо всех индексах коллекции numbres. Выполните запрос 2. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?

Добавлен индекс, вывело, что понадобилось 9 мс

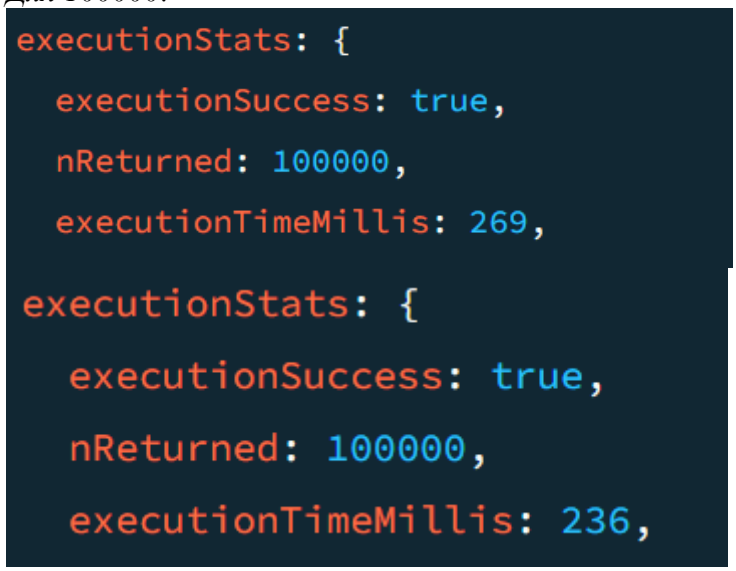
Дополнительно проведено тестирование с запросом 10000 и 10000 элементов

Для 10000:



С индексом – 27 мс, без – 190 мс

Для 100000:



Неожиданные результаты: с индексом – 269 мс, без – 236 мс

Результаты:
На маленькой выборке в 4 файла преимущества индекса почти не заметны, ускорение на 2 мс, хотя это около 20%.
На средней выборке в 10000 файлов индекс уже выигрывает значительно – ускорение в 7 раз!!!
А вот на большой индекс оказался медленнее на 33 мс или на 14%. Возможно дело в том, что был выполнен запрос на все документы из коллекции. Или что MongoDB автоматически использует collscan, когда запрашивается большая часть коллекции и нет подходящих индексов (collscan работает быстрее).

Вывод:
В ходе выполнение лабораторной работы я научился работать с MongoDB, выполнять CRUD-операции, узнать про добавление, поиск, удаление, сортировки, операторы, как связывать таблицы и индексацию. Также меня удивило, что иногда индексы могут работать медленнее, но я нашел почему так происходит.