Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет: **Инфокоммуникационных технологий**

Образовательная программа: **Мобильные и сетевые технологии**

Направление подготовки (специальность): **09.03.03 Прикладная информатика**

О Т Ч Е Т

по дисциплине «Проектирование и реализация баз данных»

на тему: Работа с БД в СУБД MongoDB

Обучающийся: Золотухин.А.Н, K32392
Преподаватель: Говорова М. М.

Дата 29.05.2023

Санкт-Петербург 2023

ВВЕДЕНИЕ

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.

1 Выполнение

Практическое задание 8.1.1:

1. Создайте базу данных learn.

```
< switched to db learn
> clear
```

2. Заполните коллекцию единорогов unicorns:

```
> db.createCollection("unicorns")
< { ok: 1 }
> 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:8
  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'});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("6474e217fd7e972f78738621")
    }
  }
learn>
```

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

```
> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
< {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
> db.unicorns.insert(document)
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("6474e27dfd7e972f78738622")
    }
  }
```

4. Проверьте содержимое коллекции с помощью метода find.

Практическое задание 8.1.2:

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

```
> db.unicorns.find({gender:"m"}).sort({name:1})
< {
    _id: ObjectId("6474e27dfd7e972f78738622"),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId("6474e217fd7e972f78738617"),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId("6474e217fd7e972f7873861d"),
    name: 'Kenny',
    loves: [
      'grape',
```

```
> db.unicorns.find({gender:"f"}).sort({name:1}).limit(3)
< {
    _id: ObjectId("6474e217fd7e972f78738618"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId("6474e217fd7e972f7873861c"),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
```

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

```
> db.unicorns.findOne({gender:"f", loves:"carrot"})
< {
    _id: ObjectId("6474e217fd7e972f78738618"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

```
> db.unicorns.find({gender:"f", loves:"carrot"}).limit(1)
< {
    _id: ObjectId("6474e217fd7e972f78738618"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

Практическое задание 8.1.3:

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

```
> db.unicorns.find({gender:"m"}, {loves:0, gender:0}).sort({name:1})
< {
    _id: ObjectId("6474e27dfd7e972f78738622"),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
  {
    _id: ObjectId("6474e217fd7e972f78738617"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
  {
    _id: ObjectId("6474e217fd7e972f7873861d"),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
  {
    _id: ObjectId("6474e217fd7e972f78738620"),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  }
```

Практическое задание 8.1.4:

1. Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find().sort({_id:-1})
< {
    _id: ObjectId("6474e27dfd7e972f78738622"),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId("6474e217fd7e972f78738621"),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
  {
    _id: ObjectId("6474e217fd7e972f78738620"),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon'
    ],
```

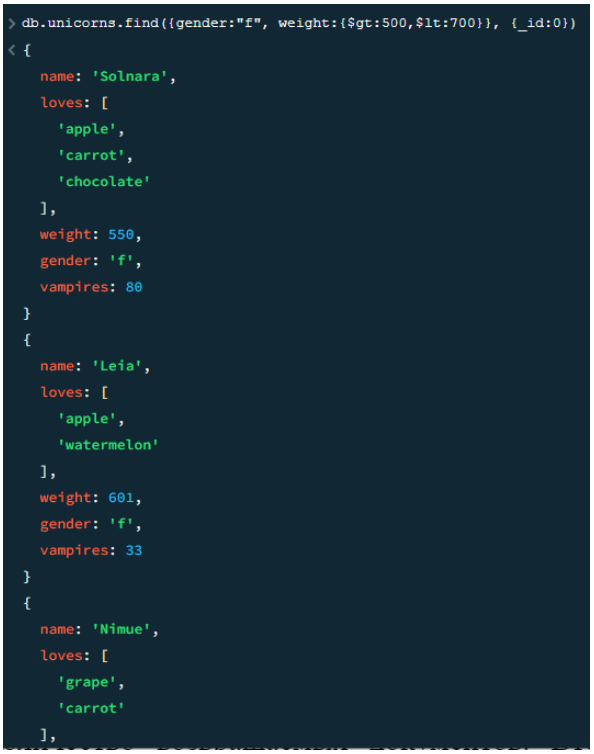
```
> db.unicorns.find().sort({$natural:-1})
< {
    _id: ObjectId("6474e27dfd7e972f78738622"),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId("6474e217fd7e972f78738621"),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
  {
    _id: ObjectId("6474e217fd7e972f78738620"),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon'
    ],
```

Практическое задание 8.1.5:

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

Практическое задание 8.1.6:

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



Практическое задание 8.1.7:

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

```
> db.unicorns.find({gender:"m", weight:{$gt:500}, loves:{$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 8.1.8:

1. Найти всех единорогов, не имеющих ключ vampires.

```
> db.unicorns.find({vampires:{$exists:false}})
< {
    _id: ObjectId("6474e217fd7e972f78738621"),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

Практическое задание 8.1.9:

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

```
> db.unicorns.find({gender:"m"}, {loves:{$slice:1}}).sort({name:1})
< {
    _id: ObjectId("6474e27dfd7e972f78738622"),
    name: 'Dunx',
    loves: [
      'grape'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId("6474e217fd7e972f78738617"),
    name: 'Horny',
    loves: [
      'carrot'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId("6474e217fd7e972f7873861d"),
    name: 'Kenny',
    loves: [
      'grape'
```

Практическое задание 8.2.1:

1. Создайте коллекцию towns, включающую следующие документы:

```
> db.createCollection("towns")
< { ok: 1 }
```

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

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

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

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

Практическое задание 8.2.2:

1. Создать функцию для вывода самцов

```
> fn=function(){return this.gender=="m";}
< [Function: fn]
```

output: null

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

```
> var cursor = db.unicorns.find(fn);null;
var cursor = db.unicorns.find(fn);null;
```

output: null

3. Вывести результат, используя forEach.

```
> cursor.forEach(function(obj) {print(obj.name);})
```

output:Dunx, Horny

4. Содержание коллекции единорогов unicorns:

Практическое задание 8.2.3: вывести количество самок единорогов весом от полутонны до 600 кг.

```
> db.unicorns.find({gender:"f", weight:{$gt:500, $lt:600}}).count()
< 2
```

Практическое задание 8.2.4: вывести список предпочтений.

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

Практическое задание 8.2.5: посчитать количество особей единорогов обоих полов.

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

Практическое задание 8.2.6:

1. Выполнить команду, Проверить содержимое коллекции unicorns.

```
> db.unicorns.replaceOne({name: 'Barny', loves: ['grape'],
  weight: 340, gender: 'm'})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
```

Практическое задание 8.2.7:

1. Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вапмиров.
2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.updateOne({name:"Ayna"}, {$set:{name:"Ayna", weight:800, gender:"f",vampires:51}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name:"Ayna"})
< {
    _id: ObjectId("6474e217fd7e972f7873861c"),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
```

Практическое задание 8.2.8:

1. Для самца единорога Raleigh внести изменения в БД: теперь он любит рэдбул.
2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.updateOne({name:"Raleigh"},{$set:{loves:["redbull"]}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name:"Raleigh"})
< {
    _id: ObjectId("6474e217fd7e972f7873861e"),
    name: 'Raleigh',
    loves: [
      'redbull'
    ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
```

Практическое задание 8.2.9:

1. Всем самцам единорогов увеличить количество убитых вапмиров на 5.
2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.updateMany({gender:"m"},{$inc:{vampires:5}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 7,
    modifiedCount: 7,
    upsertedCount: 0
  }
```

Практическое задание 8.2.10:

1. Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.
2. Проверить содержимое коллекции towns.

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

Практическое задание 8.2.11:

1. Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.
2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.updateOne({name:"Pilot"},{$push:{loves:"chocolate"}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name:"Pilot"})
< {
    _id: ObjectId("6474e217fd7e972f78738620"),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon',
      'chocolate'
    ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
```

Практическое задание 8.2.12:

1. Изменить информацию о самке единорога Aurora: теперь она любит еще и сахар, и лимоны.
2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.updateOne({name:"Aurora"},{$push:{loves:"sugar"}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name:"Aurora"})
< {
    _id: ObjectId("6474e217fd7e972f78738618"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

Практическое задание 8.2.13:

1. Создайте коллекцию towns, включающую следующие документы:

```
    insertedId: ObjectId("64762737a7b839ed0b58e0af")
  }
> db.towns.insertOne({name: "New York",
  popujatiuon: 22200000,
  last_sensus: ISODate("2009-07-31"),
  famous_for: ["status of liberty", "food"],
  mayor: {
     name: "Michael Bloomberg",
  party: "I"}}
  )
< {
    acknowledged: true,
    insertedId: ObjectId("6476274ca7b839ed0b58e0b0")
  }
> db.towns.insertOne({name: "Portland",
  popujatiuon: 528000,
  last_sensus: ISODate("2009-07-20"),
  famous_for: ["beer", "food"],
  mayor: {
     name: "Sam Adams",
  party: "D"}}
  )
< {
    acknowledged: true,
    insertedId: ObjectId("64762762a7b839ed0b58e0b1")
  }
```

2. Удалите документы с беспартийными мэрами.
3. Проверьте содержание коллекции.

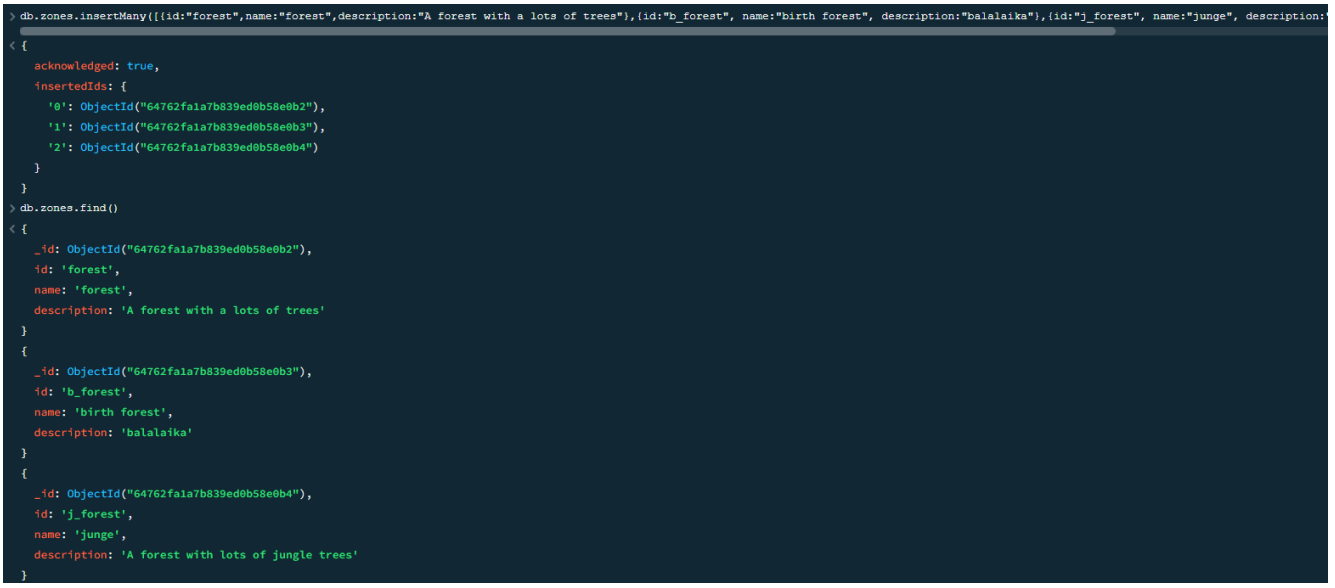
```
> db.towns.remove({"mayor.party":{$exists:false}})
< DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
< {
    acknowledged: true,
    deletedCount: 3
  }
> db.towns.find()
< {
    _id: ObjectId("6474e7c4fd7e972f78738624"),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [
      'status of liberty',
      'food'
    ],
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
  {
    _id: ObjectId("6476274ca7b839ed0b58e0b0"),
    name: 'New York',
    popujatiuon: 22200000,
```

4. Очистите коллекцию.
5. Просмотрите список доступных коллекций.

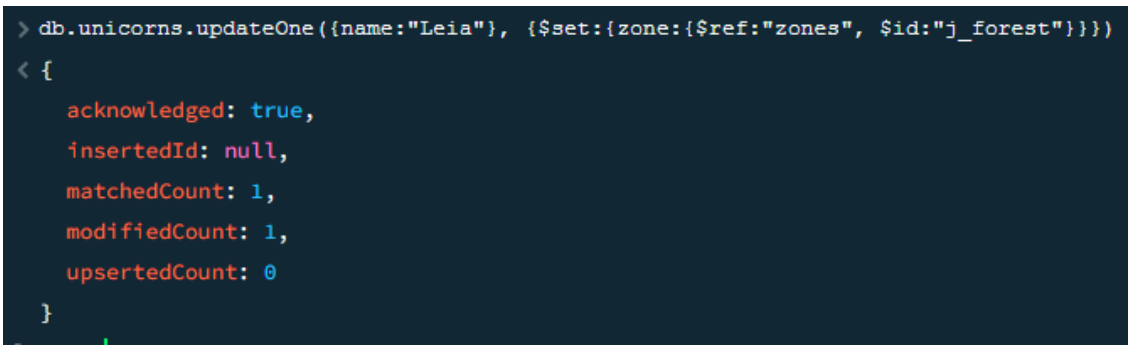
```
> db.towns.drop()
< true
> show collections
< unicorns
learn>
```

Практическое задание 8.3.1:

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

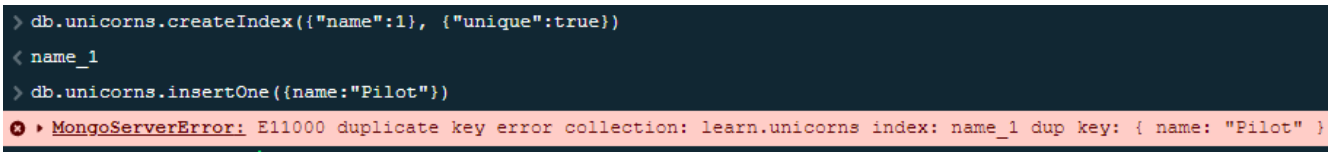


2. Включите для нескольких единорогов в документы ссылку на зону обитания, использую второй способ автоматического связывания.
3. Проверьте содержание коллекции едиорогов.



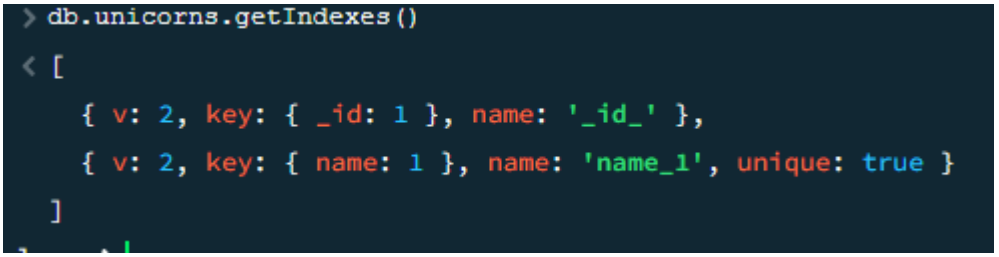
Практическое задание 8.3.2:

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



Практическое задание 8.3.3:

1. Получите информацию о всех индексах коллекции unicorns .



2. Удалите все индексы, кроме индекса для идентификатора.

```
> db.unicorns.dropIndexes("name_1")
< { nIndexesWas: 2, ok: 1 }
> db.unicorns.getIndexes()
< [ { v: 2, key: { _id: 1 }, name: '_id_' } ]
```

3. Попытайтесь удалить индекс для идентификатора.

```
> db.unicorns.dropIndexes("_id_")
MongoServerError: cannot drop _id index
```

Практическое задание 8.3.4:

1. Создайте объемную коллекцию numbers, задействовав курсор:
2. Выберите последних четыре документа.

Пометка - выполнялось почти час :)

```
> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
```

```
> db.numbers.find().sort({$natural:-1}).limit(4)
< {
    _id: ObjectId("6476325ba7b839ed0b5a6755"),
    value: 99999
  }
  {
    _id: ObjectId("6476325ba7b839ed0b5a6754"),
    value: 99998
  }
  {
    _id: ObjectId("6476325ba7b839ed0b5a6753"),
    value: 99997
  }
  {
    _id: ObjectId("6476325ba7b839ed0b5a6752"),
    value: 99996
  }
```

3. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра executionTimeMillis)
   db.numbers.explain(“executionStats”).find().sort({$natural:-1}).limit(4)

```
> db.numbers.explain("executionStats").find().sort({$natural:-1}).limit(4)
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: {
        stage: 'COLLSCAN',
        direction: 'backward'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 4,
    executionStages: {
      stage: 'LIMIT',
```

4. Создайте индекс для ключа value.
5. Получите информацию о всех индексах коллекции numbres.

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

6. Выполните запрос 2.

```
> db.numbers.explain("executionStats").find().sort({$natural:-1}).limit(4)
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'learn.numbers',
      indexFilterSet: false,
      parsedQuery: {},
      queryHash: '17830885',
      planCacheKey: '17830885',
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      winningPlan: {
        stage: 'LIMIT',
        limitAmount: 4,
        inputStage: {
          stage: 'COLLSCAN',
          direction: 'backward'
        }
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 4,
      executionTimeMillis: 0,
```

7. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса? 0
8. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?
   Время уменьшилось на 5 миллисекунд. Запрос с индексами эффективнее.

ЗАКЛЮЧЕНИЕ

В ходе работы были получены практические навыки работы с CRUD - операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, с ссылками и индексами в базе данных MongoDB.

MongoDB предоставляет мощный CLI интерфейс для выполнения CRUD операций, отличительной особенностью является интеграция полноценного языка программирования: Javascript.