Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Кафедра вычислительных систем

ОТЧЕТ
по лабораторной работе № 6 на тему:
«Работа с БД в СУБД MongoDB**»**

Выполнил: студент группы К3239
ФИО: Ермаков Максим Олегович

Проверил: преподаватель М.М. Говорова

Санкт-Петербург
2025

Цель работы

Овладеть практическими навыками работы с CRUD-операциями, вложенными объектами, агрегацией, изменением данных, ссылками и индексами в MongoDB.

Практическое задание 2.1.1

Создание базы данных и заполнение коллекции

```
> use learn
< switched to db 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('68349ba8013548781dcbd57e')
    }
  }
> db.unicorns.insertOne({
    name: "Dunx",
    loves: ["grape", "watermelon"],
    weight: 704,
    gender: "m",
    vampires: 165
  })
< {
    acknowledged: true,
    insertedId: ObjectId('68349ca4013548781dcbd57f')
  }
> db.unicorns.find()
< {
    _id: ObjectId('68349ba8013548781dcbd574'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('68349ba8013548781dcbd575'),
    name: 'Aurora',
```

Практическое задание 2.2.1

Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени. Найдите всех самок, которые любят carrot.

Ограничьте этот список первой особью с помощью функций findOne и limit

Команды MongoDB:
Вывести список самцов (gender = "m"), отсортированный по имени:

db.unicorns.find({gender: 'm'}).sort({name: 1})

Вывести список самок (gender = "f"), отсортированный по имени, ограниченный первыми 3:

db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)

Найти всех самок, которые любят "carrot":

db.unicorns.find({gender: 'f', loves: 'carrot'})

```
> db.unicorns.find({gender: 'f', loves: 'carrot'})
< {
    _id: ObjectId('68349ba8013548781dcbd575'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('68349ba8013548781dcbd578'),
    name: 'Solnara',
    loves: [
      'apple',
      'carrot',
      'chocolate'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
  {
    _id: ObjectId('68349ba8013548781dcbd57e'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
learn>
```

Ограничить этот список первой особью — способ 1 (через findOne):

db.unicorns.findOne({gender: 'f', loves: 'carrot'})

Ограничить этот список первой особью — способ 2 (через limit(1)):

db.unicorns.find({gender: "f", loves: "carrot"}).limit(1)

Практическое задание 2.2.2
Модифицируйте запрос для вывода списков самцов единорогов, исключив из результата информацию о предпочтениях и поле.

Команды MongoDB:

db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})

```
> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
< {
    _id: ObjectId('68349ba8013548781dcbd574'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
  {
    _id: ObjectId('68349ba8013548781dcbd576'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
  {
    _id: ObjectId('68349ba8013548781dcbd577'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  }
  {
    _id: ObjectId('68349ba8013548781dcbd57a'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
  {
    _id: ObjectId('68349ba8013548781dcbd57b'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  }
  {
    _id: ObjectId('68349ba8013548781dcbd57d'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  }
```

Практическое задание 2.2.3

Вывести список единорогов в обратном порядке добавления.

Команды MongoDB:

```
db.unicorns.find().sort({$natural: -1})
```

Практическое задание 2.2.4

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

Команды MongoDB:

```
db.unicorns.find({}, {loves: {$slice: 1}, _id: 0})
```

```
> 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
  }
  {
    name: 'Solnara',
    loves: [
      'apple'
    ],
    weight: 550,
```

Практическое задание 2.3.1
Вывести список самок единорогов весом от полутонны до 700 кг, исключив вывод идентификатора.

Команды MongoDB:

```
db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 700}}, {_id: 0})
```

Практическое задание 2.3.2
Вывести список самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора.

Команды MongoDB:

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

Практическое задание 2.3.3
Найти всех единорогов, не имеющих ключ vampires.

Команды MongoDB:

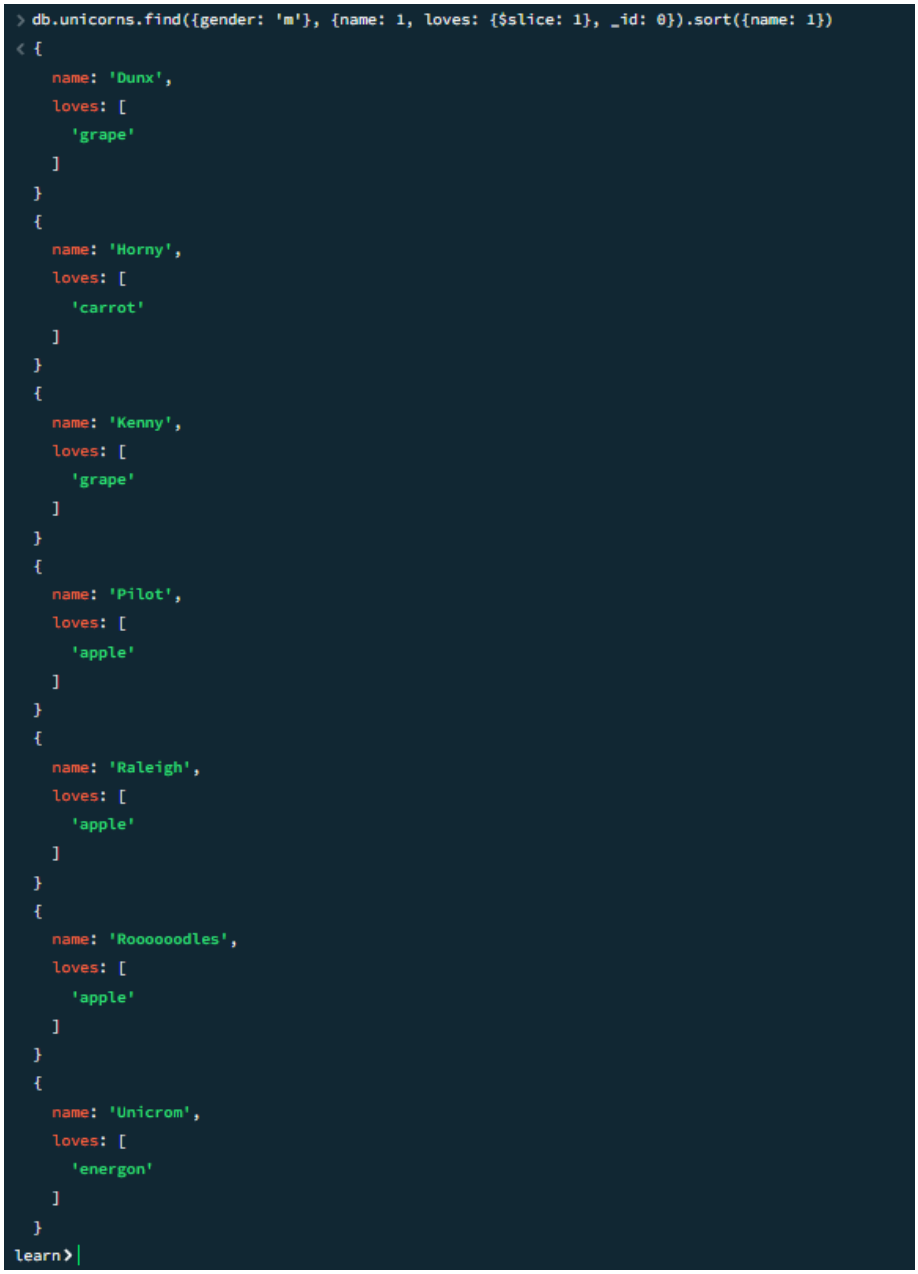
```
db.unicorns.find({vampires: {$exists: false}})
```

Практическое задание 2.3.4
Вывести упорядоченный список имен самцов единорогов с информацией об их первом предпочтении.

Команды MongoDB:

```
db.unicorns.find({gender: 'm'}, {name: 1, loves: {$slice: 1}, _id: 0}).sort({name: 1})
```

```
> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 700}}, {_id: 0})
< {
    name: 'Solnara',
    loves: [
      'apple',
      'carrot',
      'chocolate'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
  {
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
  {
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
> db.unicorns.find({gender: 'm', weight: {$gte: 500}, loves: {$all: ['grape', 'lemon']}}, {_id: 0})
< {
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
> db.unicorns.find({vampires: {$exists: false}})
< {
    _id: ObjectId('68349ba8013548781dcbd57e'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
learn>
```



Практическое задание 3.1.1

Создайте коллекцию towns и выполните выборки по мэрам с party="I" и без party.

Команды MongoDB:

db.towns.insert({...})

db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 1})

db.towns.find({"mayor.party": {$exists: false}}, {name: 1, mayor: 1})

```
> db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 1})
< {
    _id: ObjectId('6834ca0b013548781dcbd581'),
    name: 'New York',
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
> db.towns.find({"mayor.party": {$exists: false}}, {name: 1, mayor: 1})
< {
    _id: ObjectId('6834ca0b013548781dcbd580'),
    name: 'Punxsutawney',
    mayor: {
      name: 'Jim Wehrle'
    }
  }
learn>
```

Практическое задание 3.1.2
Сформировать функцию для вывода списка самцов единорогов и вывести первых двух.

Команды MongoDB:

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

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

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

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

Команды MongoDB:

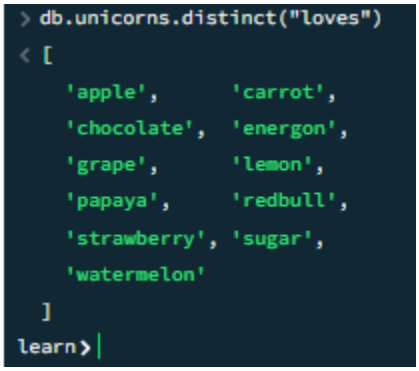
```
db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 600}}).count()
```

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

Практическое задание 3.2.2
Вывести список предпочтений.

Команды MongoDB:

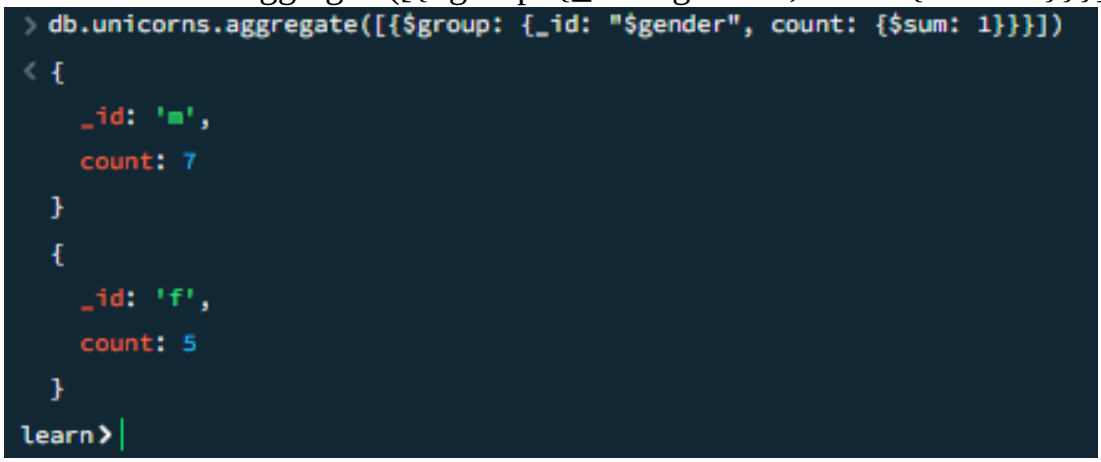
db.unicorns.distinct("loves")



Практическое задание 3.2.3
Посчитать количество особей единорогов обоих полов.

Команды MongoDB:

db.unicorns.aggregate([{$group: {_id: "$gender", count: {$sum: 1}}}])



Практическое задание 3.3.1
Добавить самца Barny.

Команды MongoDB:

db.unicorns.insertOne({name: "Barny", loves: ["grape"], weight: 340, gender: "m"})

Практическое задание 3.3.2
Обновить Ayna: вес 800, вампиры 51.

Команды MongoDB:

```
db.unicorns.update({name: "Ayna"}, {$set: {weight: 800, vampires: 51}})
```

Практическое задание 3.3.3
Обновить Raleigh: добавить redbull в loves.

Команды MongoDB:

```
db.unicorns.update({name: "Raleigh"}, {$push: {loves: "redbull"}})
```

Практическое задание 3.3.4
Увеличить количество убитых вампиров у всех самцов на 5.

Команды MongoDB:

```
db.unicorns.update({gender: "m"}, {$inc: {vampires: 5}}, {multi: true})
```

Практическое задание 3.3.5
Убрать партию у мэра Портланда.

Команды MongoDB:

```
db.towns.update({name: "Portland"}, {$unset: {"mayor.party": 1}})
```

Практическое задание 3.3.6
Обновить Pilot: добавить chocolate в loves.

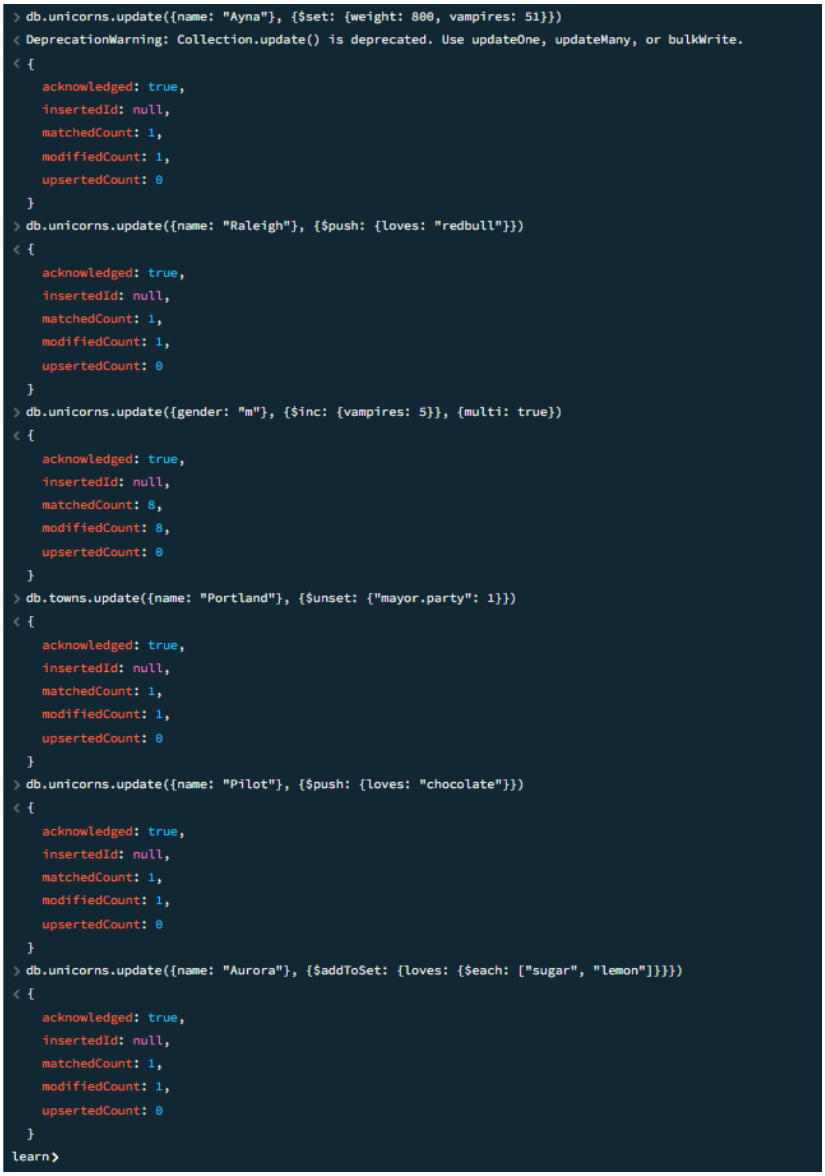
Команды MongoDB:

```
db.unicorns.update({name: "Pilot"}, {$push: {loves: "chocolate"}})
```

Практическое задание 3.3.7
Обновить Aurora: добавить sugar и lemon в loves.

Команды MongoDB:

```
db.unicorns.update({name: "Aurora"}, {$addToSet: {loves: {$each: ["sugar", "lemon"]}}})
```



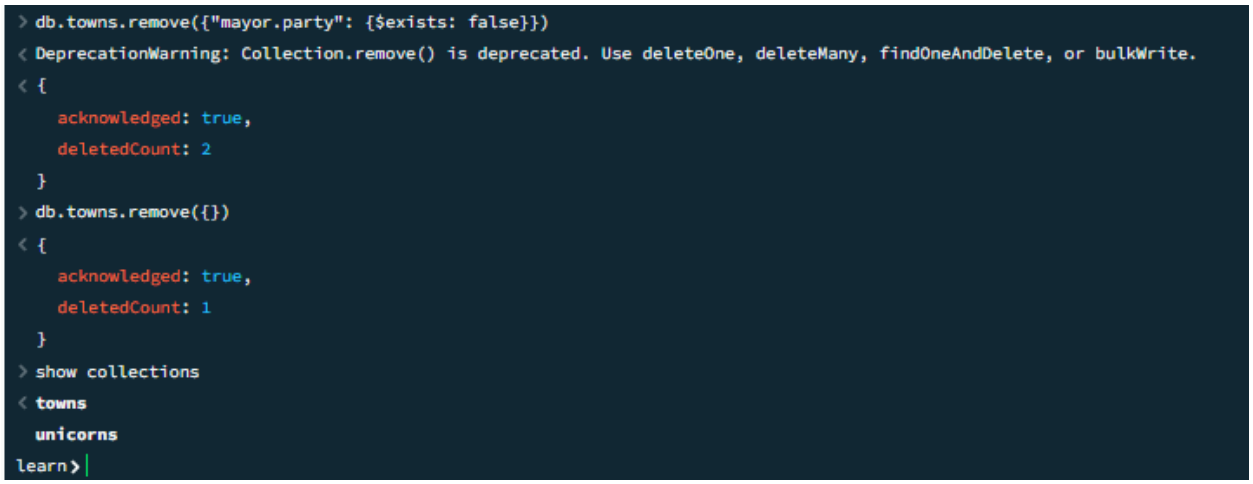
Практическое задание 3.4.1
Удалить беспартийных мэров, очистить коллекцию, просмотреть коллекции.

Команды MongoDB:

db.towns.remove({"mayor.party": {$exists: false}})

db.towns.remove({})

show collections



Практическое задание 4.1.1

Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание. Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания. Проверьте содержание коллекции единорогов.

Команды MongoDB:

db.habitats.insert({_id: "nw", name: "Northwest", desc: "Forests and rivers"});

db.habitats.insert({_id: "desert", name: "Desert", desc: "Hot and sandy"});

db.unicorns.update({name: "Horny"}, {$set: {habitat: {$ref: "habitats", $id: "nw"}}});

db.unicorns.update({name: "Aurora"}, {$set: {habitat: {$ref: "habitats", $id: "desert"}}});

db.unicorns.find();

```
> db.habitats.insert({_id: "nw", name: "Northwest", desc: "Forests and rivers"});
  db.habitats.insert({_id: "desert", name: "Desert", desc: "Hot and sandy"});
  db.unicorns.update({name: "Horny"}, {$set: {habitat: {$ref: "habitats",
  $id: "nw"}}});
  db.unicorns.update({name: "Aurora"}, {$set: {habitat: {$ref: "habitats",
  $id: "desert"}}}); db.unicorns.find();
< {
    _id: ObjectId('68349ba8013548781dcbd574'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    habitat: DBRef('habitats', 'nw')
  }
  {
    _id: ObjectId('68349ba8013548781dcbd575'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar',
      'lemon'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef('habitats', 'desert')
  }
  {
```

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

Команды MongoDB:

db.unicorns.createIndex({name: 1}, {unique: true})

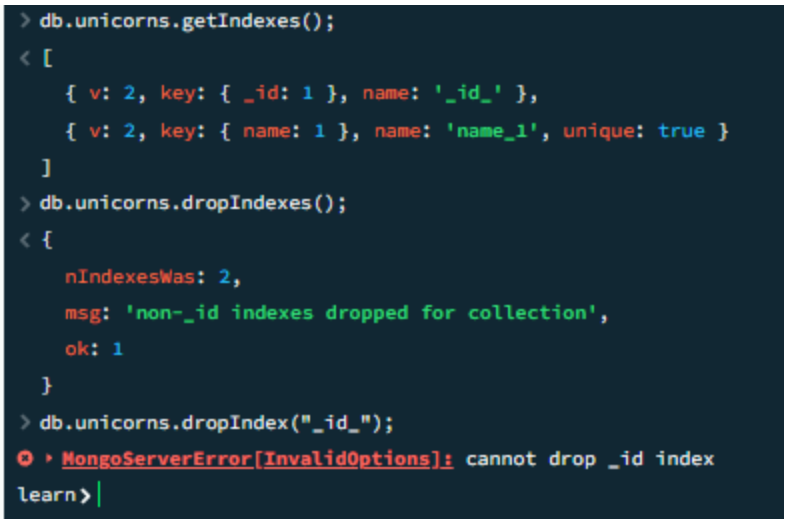
Практическое задание 4.3.1
Получите информацию о всех индексах коллекции unicorns. Удалите все индексы, кроме индекса для идентификатора. Попытайтесь удалить индекс для идентификатора.

Команды MongoDB:

db.unicorns.getIndexes();

db.unicorns.dropIndexes();

db.unicorns.dropIndex("_id_");



Практическое задание 4.4.1

Создайте объемную коллекцию numbers, задействовав курсор. Выберите последние четыре документа. Проанализируйте план выполнения запроса. Сколько потребовалось времени на выполнение запроса? Создайте индекс для ключа value. Получите информацию о всех индексах коллекции numbers. Выполните запрос 2. Проанализируйте план выполнения запроса с установленным индексом. Сравните время выполнения запросов с индексом и без.

Команды MongoDB:

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

db.numbers.find().sort({$natural: -1}).limit(4)

db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)

db.numbers.createIndex({value: 1})

db.numbers.getIndexes()

db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
```

Результат с индексом:

```
> 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' }
  ]
> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'learn.numbers',
      parsedQuery: {},
      indexFilterSet: false,
      queryHash: '8F2383EE',
      planCacheShapeHash: '8F2383EE',
      planCacheKey: '7DF350EE',
      optimizationTimeMillis: 0,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      prunedSimilarIndexes: false,
      winningPlan: {
        isCached: false,
        stage: 'LIMIT',
        limitAmount: 4,
        inputStage: {
          stage: 'COLLSCAN',
          direction: 'backward'
        }
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 4,
      executionTimeMillis: 0,
      totalKeysExamined: 0,
      totalDocsExamined: 4,
      executionStages: {
        isCached: false,
        stage: 'LIMIT',
        nReturned: 4,
        executionTimeMillisEstimate: 0,
```

Результат без индекса:

```
> db.numbers.find().sort({$natural: -1}).limit(4)
< {
    _id: ObjectId('6834daf9013548781dcd5c23'),
    value: 99999
  }
  {
    _id: ObjectId('6834daf9013548781dcd5c22'),
    value: 99998
  }
  {
    _id: ObjectId('6834daf9013548781dcd5c21'),
    value: 99997
  }
  {
    _id: ObjectId('6834daf9013548781dcd5c20'),
    value: 99996
  }
> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'learn.numbers',
      parsedQuery: {},
      indexFilterSet: false,
      queryHash: '8F2383EE',
      planCacheShapeHash: '8F2383EE',
      planCacheKey: '7DF350EE',
      optimizationTimeMillis: 0,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      prunedSimilarIndexes: false,
      winningPlan: {
        isCached: false,
        stage: 'LIMIT',
        limitAmount: 4,
        inputStage: {
          stage: 'COLLSCAN',
          direction: 'backward'
        }
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 4,
      executionTimeMillis: 1,
      totalKeysExamined: 0,
      totalDocsExamined: 4,
      executionStages: {
        isCached: false,
        stage: 'LIMIT',
        nReturned: 4,
        executionTimeMillisEstimate: 0,
        works: 5,
        advanced: 4,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
```

Без индекса происходит полный просмотр коллекции, который занял 1мс executionTimeMillis, а с индексом происходит индексный поиск, который занял менее около 0мс.

Выводы

В ходе выполнения лабораторной работы были изучены основные возможности MongoDB: вставка, выборка, изменение и удаление документов (CRUD), работа с вложенными документами, агрегация данных, создание и использование индексов, а также связи между коллекциями. Полученные навыки позволяют эффективно использовать MongoDB для хранения и обработки данных в реальных приложениях.