Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

Лабораторная работа №6

Выполнил:
Конопля А. К.
Проверила:
Говорова М. М.

Санкт-Петербург
2023

Введение

Данная лабораторная работа посвящена изучению mongod.

Цель

Целью данной лабораторной работы является написание процедур и триггеров логирования изменения данных в базе.

Задачи

Практическое задание 2.1.1:

1. Создайте базу данных learn.
2. Заполните коллекцию единорогов unicorns
3. Используя второй способ, вставьте в коллекцию единорогов документ
4. Проверьте содержимое коллекции с помощью метода find.

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.

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

Практическое задание 2.1.4:

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

Практическое задание 2.3.1:

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

Практическое задание 2.3.2:

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

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ `vampires`.

Практическое задание 2.3.4:

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

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 беспартийных мэров (`party` отсутствует). 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.

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

Практическое задание 3.2.2:

Вывести список предпочтений.

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

Практическое задание 3.3.1:

1. Выполнить команду:

```
> db.unicorns.save({name: 'Barny', loves: ['grape'],
weight: 340, gender: 'm'})
```

2. Проверить содержимое коллекции `unicorns`.

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`.

Практическое задание 3.3.3:

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

Практическое задание 3.3.4:

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

Практическое задание 3.3.5:

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

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`.

Практическое задание 3.4.1:

1. Создайте коллекцию towns, включающую следующие документы:
2. Удалите документы с беспартийными мэрами.
3. Проверьте содержание коллекции.
4. Очистите коллекцию.
5. Просмотрите список доступных коллекций.

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, использую второй способ автоматического связывания.
3. Проверьте содержание коллекции едиорогов.

Практическое задание 4.2.1:

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

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.

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

Задание 2.1.1

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264b'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264c'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264d'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264e'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264f'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202650'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202651'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202652'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202653'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('65eee9aab6b1ffa60d202654'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65eee9aab6b1ffa60d202655'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Содержание таблицы unicorns

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: "m"}).sort({name: 1})
[
  {
    _id: ObjectId('65eee9eab6b1ffa60d202656'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264b'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202651'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65eee9aab6b1ffa60d202654'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202652'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264e'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264d'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

Список самцов, отсортированный по имени

```
learn> db.unicorns.find({gender: "f"}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264c'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202650'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202653'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

Список самок, ограниченный тремя особями и отсортированный по имени

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

```
learn> db.unicorns.find({gender: "f", loves: "carrot"}).sort({name: 1}).limit(1)
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264c'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Вывод данных, ограниченный limit()

```
learn> db.unicorns.findOne({gender: "f", loves: "carrot"})
{
  _id: ObjectId('65eee99fb6b1ffa60d20264c'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Вывод данных ограниченный finOne()

Задание 2.2.2

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

```
learn> db.unicorns.find({gender: "m"}, {loves: 0, gender: 0})
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264b'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264d'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264e'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202651'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202652'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('65eee9aab6b1ffa60d202654'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('65eee9eab6b1ffa60d202656'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
]
```

Модифицированный запрос на вывод

Задание 2.2.3

Вывести единорогов в обратном порядке добавления

Список единорогов в обратном порядке от добавления

Задание 2.2.4

Вывести список единорогов с названием первого любимого предпочтения, исключив идентификатор

```
learn> db.unicorns.find({}, {name: 1, firstLove: {$arrayElemAt: ["$loves", 0]}})
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264b'),
    name: 'Horny',
    firstLove: 'carrot'
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264c'),
    name: 'Aurora',
    firstLove: 'carrot'
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264d'),
    name: 'Unicrom',
    firstLove: 'energon'
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264e'),
    name: 'Roooooodles',
    firstLove: 'apple'
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264f'),
    name: 'Solnara',
    firstLove: 'apple'
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202650'),
    name: 'Ayna',
    firstLove: 'strawberry'
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202651'),
    name: 'Kenny',
    firstLove: 'grape'
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202652'),
    name: 'Raleigh',
    firstLove: 'apple'
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202653'),
    name: 'Leia',
    firstLove: 'apple'
  },
  {
    _id: ObjectId('65eee9aab6b1ffa60d202654'),
    name: 'Pilot',
    firstLove: 'apple'
  },
  {
    _id: ObjectId('65eee9aab6b1ffa60d202655'),
    name: 'Nimue',
    firstLove: 'grape'
  },
  {
    _id: ObjectId('65eee9eab6b1ffa60d202656'),
    name: 'Dunx',
    firstLove: 'grape'
  }
]
```

команда и результат выполнения команды

Задание 2.3.1

Вывести список самок с весом от 500 до 700 кг

```
learn> db.unicorns.find({gender: "f", weight: {$gt : 500, $lt: 700}})
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264f'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202653'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('65eee9aab6b1ffa60d202655'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Запрос с использованием $lt и $gt

Во время выполнения задания я столкнулся с проблемой, оператор $in не выводил никаких данных

```
learn> db.unicorns.find({weight: {$in : [500, 700]}, gender: "f"}, {})
```

Запрос с использованием оператора $in

Задание 2.3.2

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

```
learn> db.unicorns.find({gender: "m", weight: {$gt: 500}, loves: {$exists: ["grape", "lemon"]}})
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264b'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264d'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264e'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('65eee99fb6b1ffa60d202651'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65eee9aab6b1ffa60d202654'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65eee9aab6b1ffa60d202656'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
```

Код запроса и результат выоплнения

Задние 2.3.3

Найти всех единорогов, не имеющих ключ `vampires`.

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('65eee9aab6b1ffa60d202655'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Задание 2.3.4

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

Код запроса и результат выоплнения

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.*

1

```
learn> db.towns.insertMany([
...   {
...     "name": "Punxsutawney",
...     "population": 6200,
...     "last_sensus": ISODate("2008-01-31"),
...     "famous_for": [],
...     "mayor": {
...       "name": "Jim Wehrle"
...     }
...   },
...   {
...     "name": "New York",
...     "population": 22200000,
...     "last_sensus": ISODate("2009-07-31"),
...     "famous_for": ["status 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"
...     }
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65f81cb5b6b1ffa60d202658'),
    '1': ObjectId('65f81cb5b6b1ffa60d202659'),
    '2': ObjectId('65f81cb5b6b1ffa60d20265a')
  }
}
```

2

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

3

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

```
learn> function printMaleUnicorns() {
...     const maleUnicorns = db.unicorns.find({ gender: "m" },{_id: 0, name: 1, weight: 1}).sort({ name: 1 }).limit(2);
...     maleUnicorns.forEach(function(unicorn){print("Name: " + unicorn.name + ", Weight: "+ unicorn.weight);});
... }
[Function: printMaleUnicorns]
learn> printMaleUnicorns()
Name: Dunx, Weight: 704
Name: Horny, Weight: 600
```

Ввод функции в консоль и вывод работы фукнции

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Была использована функция countDocuments, так как Collections.count – устаревшая команда, которая будет удалена в ближайших обновлениях mongodb

Практическое задание 3.2.2:

Вывести список предпочтений.

```
learn> db.unicorns.aggregate([{ $unwind: "$loves" }, { $group: { _id: null, preferences: { $addToSet: "$loves" } } }, { $project: { _id: 0, preferences: 1 } }]);
[
  {
    preferences: [
      'lemon',      'energon',
      'carrot',     'chocolate',
      'redbull',    'grape',
      'strawberry', 'sugar',
      'watermelon', 'apple',
      'papaya'
    ]
  }
]
```

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

```
learn> 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`.

1.

```
learn> db.unicorns.insert({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6603ff61b6b1ffa60d20265b') }
}
```

Была использована команда insert, так как команда save подходит только для mongodb версии ниже 4.0.

2.

```
[
  {
    _id: ObjectId('6603ff61b6b1ffa60d20265b'),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
```

Единорог Barny был добавлен в коллекцию

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`.

1.

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

2.

```
learn> db.unicorns.find({name: "Ayna"})
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d202650'),
    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`.

1.

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

2.

```
learn> db.unicorns.find({name: "Raleigh"})
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d202652'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', '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`.

1.

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

Запрос на увеличение атрибута vampires

2.

```
learn> db.unicorns.find({name: "Raleigh"})
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d202652'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Пример данных до запроса

```
learn> db.unicorns.find({name: "Raleigh"})
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d202652'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Пример данных после запроса

Практическое задание 3.3.5:

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

1.

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

Запрос на изменений данных

```
{
  _id: ObjectId('65f81cb5b6b1ffa60d20265a'),
  name: 'Portland',
  population: 528000,
  last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
  famous_for: [ 'beer', 'food' ],
  mayor: { name: 'Sam Adams', party: null }
}
```

Обновлённые данные о городе Portland

Практическое задание 3.3.6:

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

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

Запрос на обновление данных

2. *Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.find({name: "Pilot"})
[
  {
    _id: ObjectId('65eee9aab6b1ffa60d202654'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

Обновлённые данные

Практическое задание 3.3.7:

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

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

2. *Проверить содержимое коллекции* `unicorns`.

```
learn> db.unicorns.find({name: "Aurora"})
[
  {
    _id: ObjectId('65eee99fb6b1ffa60d20264c'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 3.4.1:

1. *Создайте коллекцию towns*

```
learn> db.towns.find()
[
  {
    _id: ObjectId('65f81cb5b6b1ffa60d202658'),
    name: 'Punxsutawney',
    population: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('65f81cb5b6b1ffa60d202659'),
    name: 'New York',
    population: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('65f81cb5b6b1ffa60d20265a'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

2. *Удалите документы с беспартийными мэрами.*

```
learn> db.towns.deleteMany({ "mayor.party": null });
{ acknowledged: true, deletedCount: 1 }
```

3. *Проверьте содержание коллекции.*

```
learn> db.towns.find()
[
  {
    _id: ObjectId('65f81cb5b6b1ffa60d202659'),
    name: 'New York',
    population: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('65f81cb5b6b1ffa60d20265a'),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

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
users
```

Контрольные вопросы

1) *Как используется оператор точка?*

   Оператор точка используется в mongodb для получения информации из вложенных объектов

   Например :

```
mayor: { name: 'Sam Adams', party: 'D' }
```

2) *Как можно использовать курсов?*

   Курсор в mongodb используется для последовательного чтения каждого документа из результата запроса

3) *Какие возможности агрегирования данных существуюи в mongodb?*

   $group: группировка по атрибутам

   $unwind: используется для “разматывания” документов, использующих массивы

   $project: используется для использования нескольких специальных из коллекции

   $sort: сортировка

   $skip: пропуск документов в имеющемся множестве документов

   $limit: ограничение количества документов для вывода

4) Какая из функций save или update более детально позволит настроить редактирование документов коллекции?
   Update позволяет меня уже имеющиеся данные в таблицу, а также походит для выполнения для операций над множеством документов, в то время как save в форматце обновления данных перезаписывает объект по _id, что не так детально для обновления данных, как update. (save устаревшая команда, которая будет удалена в ближайших обновлениях)

5) Как происходит удаление документов из коллекции по умолчанию?
В mongodb есть deleteOne() и deleteMany(), по умолчанию первая команда удалит первый элемент в коллекции, а deleteMany({}) удалит все записи в коллекции.

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.unicorn_habitats.insert({_id: "forest", zone_name:"forest", full_name: "magic forest", description: ""})
```

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

```
learn> db.unicorns.update({name: "Pilot"}, {$set: {"habitat_zone": {$ref: "unicorn_habitats", $id: "forest"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: "Pilot"})
[
  {
    _id: ObjectId('65eee9aab6b1ffa60d202654'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59,
    habitat_zone: DBRef('unicorn_habitats', 'forest')
  }
]
learn> db.unicorns.update({name: "Aurora"}, {$set: {"habitat_zone": {$ref: "unicorn_habitats", $id: "forest"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorn_habitats.find()
[
  {
    _id: 'forest',
    zone_name: 'forest',
    full_name: 'magic forest',
    description: ''
  }
]
```

Практическое задание 4.2.1:

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

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 }
]
```

2. *Удалите все индексы, кроме индекса для идентификатора.*

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

3. *Попытайтесь удалить индекс для идентификатора.*

```
learn> 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})}
```

```
learn> for (var i = 0; i < 100000; i++) { db.numbers.insert({ value: i }); };
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6604164fb6b1ffa60d236b8f') }
}
learn> db.numbers.find().skip(90000)
[
  { _id: ObjectId('6604164bb6b1ffa60d234480'), value: 90000 },
  { _id: ObjectId('6604164bb6b1ffa60d234481'), value: 90001 },
  { _id: ObjectId('6604164cb6b1ffa60d234482'), value: 90002 },
  { _id: ObjectId('6604164cb6b1ffa60d234483'), value: 90003 },
  { _id: ObjectId('6604164cb6b1ffa60d234484'), value: 90004 },
  { _id: ObjectId('6604164cb6b1ffa60d234485'), value: 90005 },
  { _id: ObjectId('6604164cb6b1ffa60d234486'), value: 90006 },
  { _id: ObjectId('6604164cb6b1ffa60d234487'), value: 90007 },
  { _id: ObjectId('6604164cb6b1ffa60d234488'), value: 90008 },
  { _id: ObjectId('6604164cb6b1ffa60d234489'), value: 90009 },
  { _id: ObjectId('6604164cb6b1ffa60d23448a'), value: 90010 },
  { _id: ObjectId('6604164cb6b1ffa60d23448b'), value: 90011 },
  { _id: ObjectId('6604164cb6b1ffa60d23448c'), value: 90012 },
  { _id: ObjectId('6604164cb6b1ffa60d23448d'), value: 90013 },
  { _id: ObjectId('6604164cb6b1ffa60d23448e'), value: 90014 },
  { _id: ObjectId('6604164cb6b1ffa60d23448f'), value: 90015 },
  { _id: ObjectId('6604164cb6b1ffa60d234490'), value: 90016 },
  { _id: ObjectId('6604164cb6b1ffa60d234491'), value: 90017 },
  { _id: ObjectId('6604164cb6b1ffa60d234492'), value: 90018 },
  { _id: ObjectId('6604164cb6b1ffa60d234493'), value: 90019 }
]
Type "it" for more
```

2. *Выберите последних четыре документа.*

```
learn> db.numbers.find().sort({_id: -1}).limit(4)
[
  { _id: ObjectId('6604164fb6b1ffa60d236b8f'), value: 99999 },
  { _id: ObjectId('6604164fb6b1ffa60d236b8e'), value: 99998 },
  { _id: ObjectId('6604164fb6b1ffa60d236b8d'), value: 99997 },
  { _id: ObjectId('6604164fb6b1ffa60d236b8c'), value: 99996 }
]
```

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

```
learn> db.numbers.find().sort({ _id: -1 }).limit(4).explain("executionStats");
```

```
executionTimeMillis: 8,
```

4. *Создайте индекс для ключа* `value`*.*

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

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' }
]
```

6. *Выполните запрос 2.*

```
learn> db.numbers.find().sort({_id: -1}).limit(4)
[
  { _id: ObjectId('6604164fb6b1ffa60d236b8f'), value: 99999 },
  { _id: ObjectId('6604164fb6b1ffa60d236b8e'), value: 99998 },
  { _id: ObjectId('6604164fb6b1ffa60d236b8d'), value: 99997 },
  { _id: ObjectId('6604164fb6b1ffa60d236b8c'), value: 99996 }
]
```

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

```
executionTimeMillis: 1,
```

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

Время выполнения запроса без индекса 8 мс.

Время выполенния запроса с индексом по values 1 мс

Запрос с индексом эффективнее.

Контрольные вопросы:

1) Назовите способы связывания коллекций в MongoDB.

   Ручная связка

   Ручная установка ссылок сводится к присвоению значения поля _id одного документа полю другого документа.

   Автоматическая связка

   Используя функциональность DBRef, мы можем установить автоматическое связывание между документами. При добавлении нового документа генерирует _id, который мы можем получить с помощью свойства insertedId результата функции.

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

   В MongoDB можно установить максимум 64 индекса на одну коллекцию. Это ограничение включает в себя все типы индексов: одно поле, составные, текстовые, геопространственные и т.д.

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

   db.collection.getIndexes();

Вывод

В ходу выполнения работы был изучен mongodb. Изучено создание и управление базой данных MongoDB: новой базы данных "learn" и понимание ее структуры. Формирование коллекций для организации данных. Освоены основные операции CRUD: Заполнение коллекции "unicorns" документами, включая создание, чтение, обновление и удаление данных. Применение методов вставки, поиска, обновления и удаления документов. Изучены и применены операции фильтрации и сортировки: Фильтрация данных по различным критериям, таким как пол, предпочтения, вес и другие характеристики. Сортировка результатов запросов по заданным полям, например, по имени или весу. Работа с индексами для улучшения производительности: Создание индексов для оптимизации запросов к базе данных. Удаление и управление индексами для обеспечения эффективной работы с данными. Изучено связывание данных и агрегация: Создание ссылок между документами разных коллекций, например, между единорогами и их зонами обитания. Применение агрегации для выполнения сложных операций над данными, таких как группировка, фильтрация и вычисления. Получены навыки работы с курсорами и обходом результатов: Использование курсоров для работы с большими объемами данных. Обход результатов запросов с помощью циклов и методов, таких как forEach. Лабораторная работа с MongoDB позволила понять основы работы с NoSQL базой данных, изучить возможности CRUD операций, индексирования данных, агрегации и другие важные аспекты работы с базой данных. Полученные навыки и опыт могут быть использованы в разработке приложений, требующих эффективное хранение и управление данными.

Список источников:

1. MongoDB CRUD Operations [Электронный ресурс] // mongoDB. Documentation: официальный сайт MongoDB. URL: https://docs.mongodb.com/manual/ (дата обращения: 02.05.2023).
2. MongoDB – Краткое руководство [Электронный ресурс] // CoderLessons.com. Уроки по программированию, DevOps и другим IT-технологиям: сайт, 2019. URL: https://coderlessons.com/tutorials/bazy-dannykh/uchitsia-mongodb/mongodb-kratkoe-rukovodstvo (дата обращения: 02.05.2023).
3. Кайл Б. MongoDB в действии [Электронный ресурс] // Доступ в ЭБС «Лань». Режим доступа: https://e.lanbook.com/book/4156 (дата обращения: 05.05.2023).
4. Онлайн-руководство по MongoDB [Электронный ресурс] // METANIT.COM. Сайт о программировании. URL: https://metanit.com/nosql/mongodb/ (дата обращения: 05.05.2023).