Национальный исследовательский университет ИТМО

Лабораторная работа №5

«Работа с БД в СУБД MongoDB»

По дисциплине
«Проектирование и реализация баз данных»

Выполнил:
Кривцов П.А.
Группа:
K3240
Преподаватель:
Говорова М.М.

Санкт-Петербург
2022 г

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.

ВЫПОЛНЕНИЕ

8.1.1

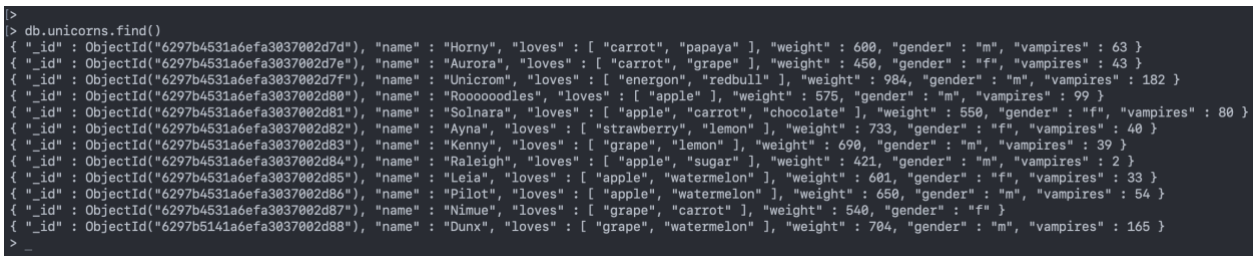
- Создадим базу данных learn и заполним коллекцию единорогов unicorns:

```
> use learn
switched to db learn
> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690,  gender: 'm', vampires: 39});
WriteResult({ "nInserted" : 1 })
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: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
WriteResult({ "nInserted" : 1 })
> _
```

- Используя второй способ, вставим 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)
WriteResult({ "nInserted" : 1 })
> _
```

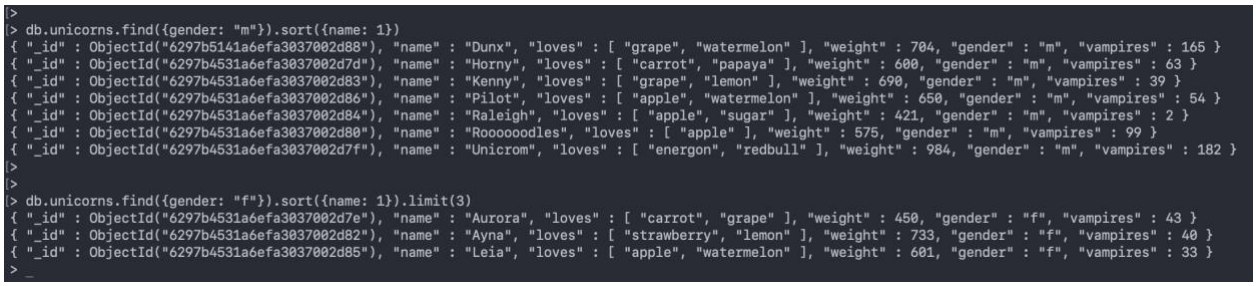
- Проверим содержимое базы с помощью метода find



8.1.2

- Сформируем запросы для вывода списков самцов и самок единорогов. Ограничим список самок первыми тремя особями. Отсортируем списки по имени.

  Самцы: *db.unicorns.find({gender: "m"}).sort({name: 1})*
  Самки: *db.unicorns.find({gender: "f"}).sort({name: 1}).limit(3)*



- Найдем всех самок, которые любят carrot. Ограничим этот список первой особью.

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

8.1.3

- Модифицируем запрос для вывода списка самцов единорогов, исключив из результата информацию о предпочтениях и поле

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

```
[>
[> db.unicorns.find({gender: "m"}, {loves: 0, gender: 0})
{ "_id" : ObjectId("6297b4531a6efa3037002d7d"), "name" : "Horny", "weight" : 600, "vampires" : 63 }
{ "_id" : ObjectId("6297b4531a6efa3037002d7f"), "name" : "Unicrom", "weight" : 984, "vampires" : 182 }
{ "_id" : ObjectId("6297b4531a6efa3037002d80"), "name" : "Roooooodles", "weight" : 575, "vampires" : 99 }
{ "_id" : ObjectId("6297b4531a6efa3037002d83"), "name" : "Kenny", "weight" : 690, "vampires" : 39 }
{ "_id" : ObjectId("6297b4531a6efa3037002d84"), "name" : "Raleigh", "weight" : 421, "vampires" : 2 }
{ "_id" : ObjectId("6297b4531a6efa3037002d86"), "name" : "Pilot", "weight" : 650, "vampires" : 54 }
{ "_id" : ObjectId("6297b5141a6efa3037002d88"), "name" : "Dunx", "weight" : 704, "vampires" : 165 }
> _
```

8.1.4

- Выведем список единорогов в обратном порядке добавления

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

```
> db.unicorns.find().sort({ $natural: -1 })
{ "_id" : ObjectId("6297b5141a6efa3037002d88"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
{ "_id" : ObjectId("6297b4531a6efa3037002d87"), "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
{ "_id" : ObjectId("6297b4531a6efa3037002d86"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("6297b4531a6efa3037002d85"), "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("6297b4531a6efa3037002d84"), "name" : "Raleigh", "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "_id" : ObjectId("6297b4531a6efa3037002d83"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "_id" : ObjectId("6297b4531a6efa3037002d82"), "name" : "Ayna", "loves" : [ "strawberry", "lemon" ], "weight" : 733, "gender" : "f", "vampires" : 40 }
{ "_id" : ObjectId("6297b4531a6efa3037002d81"), "name" : "Solnara", "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("6297b4531a6efa3037002d80"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 99 }
{ "_id" : ObjectId("6297b4531a6efa3037002d7f"), "name" : "Unicrom", "loves" : [ "energon", "redbull" ], "weight" : 984, "gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("6297b4531a6efa3037002d7e"), "name" : "Aurora", "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" : "f", "vampires" : 43 }
{ "_id" : ObjectId("6297b4531a6efa3037002d7d"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" : 63 }
> _
```

8.1.5

- Выведем список единорогов с названием первого любимого предпочтения, исключив идентификатор

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

```
[> db.unicorns.find({}, {loves: {$slice: 1}, _id: false})
{ "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, "gender" : "f", "vampires" : 80 }
{ "name" : "Ayna", "loves" : [ "strawberry" ], "weight" : 733, "gender" : "f", "vampires" : 40 }
{ "name" : "Kenny", "loves" : [ "grape" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "name" : "Raleigh", "loves" : [ "apple" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "name" : "Leia", "loves" : [ "apple" ], "weight" : 601, "gender" : "f", "vampires" : 33 }
{ "name" : "Pilot", "loves" : [ "apple" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "name" : "Nimue", "loves" : [ "grape" ], "weight" : 540, "gender" : "f" }
{ "name" : "Dunx", "loves" : [ "grape" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
> _
```

8.1.6

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

db.unicorns.find({weight: {$gt: 500, $lt: 700}}, {_id: false})

```
> db.unicorns.find({weight: {$gt: 500, $lt: 700}}, {_id: false})
{ "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 }
{ "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender" : "f", "vampires" : 33 }
{ "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
> _
```

8.1.7

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

db.unicorns.find({gender: "m", weight: {$gt: 500}, loves: {$all: ["grape", "lemon"]}}, {_id: false})

```
> db.unicorns.find({gender: "m", weight: {$gt: 500}, loves: {$all: ["grape", "lemon"]}}, {_id: false})
{ "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
> _
```

8.1.8

- Найдем всех единорогов, не имеющих ключ vampires.

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

```
> db.unicorns.find({vampires: {$exists: false}})
{ "_id" : ObjectId("6297b4531a6efa3037002d87"), "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
> _
```

8.1.9

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

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

```
> db.unicorns.find({gender: "m"}, {loves: {$slice: 1}, name: 1, _id: 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" ] }
> _
```

8.2.1

- Создадим коллекцию towns

```
[>
 > db.towns.insert({name: "Punxsutawney ",
 ... populatiuon: 6200,
 ... last_sensus: ISODate("2008-01-31"),
 ... famous_for: [""],
 ... mayor: {
 ...     name: "Jim Wehrle"
 ... }}
[... )
 WriteResult({ "nInserted" : 1 })
[>
 > db.towns.insert({name: "New York",
 ... populatiuon: 22200000,
 ... last_sensus: ISODate("2009-07-31"),
 ... famous_for: ["status of liberty", "food"],
 ... mayor: {
 ...     name: "Michael Bloomberg",
 ... party: "I"}}
[... )
 WriteResult({ "nInserted" : 1 })
[>
 > db.towns.insert({name: "Portland",
 ... populatiuon: 528000,
 ... last_sensus: ISODate("2009-07-20"),
 ... famous_for: ["beer", "food"],
 ... mayor: {
 ...     name: "Sam Adams",
 ... party: "D"}}
[... )
 WriteResult({ "nInserted" : 1 })
 > _
```

- Сформируем 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})

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

- Сформируем 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})

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

8.2.2

- Сформируем функцию для вывода списка самцов единорогов

  get_male = function() { return this.gender == "m"; }

```
[>
[> get_male = function() { return this.gender == "m"; }
function() { return this.gender == "m"; }
> _
```

- Создадим курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.

```
[>
[> var males_cursor = db.unicorns.find(get_male); null;
null
[> males_cursor.limit(2).sort({name: 1}); null;
null
> _
```

- Выведем результат, используя forEach

```
[>
[> males_cursor.forEach(function(element) { print(element.name); })
Dunx
Horny
> _
```

8.2.3

- Выведем количество самок единорогов весом от полутоны до 600 кг.

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

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

8.2.4

- Выведем список предпочтений, которые хотя бы раз появляются в таблице

db.unicorns.distinct("loves")

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

8.2.5

- Посчитаем количество особей единорогов обоих полов *db.unicorns.aggregate({"$group": {_id: "$gender", count: {"$sum": 1}}})*

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

8.2.6

- Выполним команду *db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})*

```
[>
> db.unicorns.save({name: 'Barny', loves: ['grape'],
... weight: 340, gender: 'm'})
WriteResult({ "nInserted" : 1 })
> _
```

- В коллекции *unicorns* стало на один документ больше

8.2.7

- Посмотрим на информацию об особи с именем Ayna

```
[>
[> db.unicorns.find({name: "Ayna"})
{ "_id" : ObjectId("6297b4531a6efa3037002d82"), "name" : "Ayna", "loves" : [ "strawberry", "lemon" ], "weight" : 733, "gender" : "f", "vampires" : 40 }
[>
> _
```

- Изменим для нее поле weight на 600, а vampires на 51 *db.unicorns.update({name: "Ayna"}, {weight: 600, vampires: 51})*

```
[>
[> db.unicorns.update({name: "Ayna"}, {name: "Ayna", weight: 600, vampires: 51})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> _
```

- Посмотрим теперь на данные для Ayna:

```
[>
[> db.unicorns.find({name: "Ayna"})
{ "_id" : ObjectId("6297b4531a6efa3037002d82"), "name" : "Ayna", "weight" : 600, "vampires" : 51 }
> _
```

Видим, что поля, не участвующие в запросе update, были удалены

8.2.8

- Найдем самца с именем Raleigh

```
[>
[> db.unicorns.find({name: "Raleigh"})
{ "_id" : ObjectId("6297b4531a6efa3037002d84"), "name" : "Raleigh", "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
> _
```

- Внесем изменения в БД: теперь он любит рэдбул.
  db.unicorns.update({name: "Raleigh"}, {$set: {loves: ["rebull"]}})

```
[>
[> db.unicorns.update({name: "Raleigh"}, {$set: {loves: ["rebull"]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> _
```

- Посмотрим на этот документ после изменения

```
[>
[> db.unicorns.find({name: "Raleigh"})
{ "_id" : ObjectId("6297b4531a6efa3037002d84"), "name" : "Raleigh", "loves" : [ "rebull" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
> _
```

Заметим, что обновилось только одно поле, остальные остались.

8.2.9

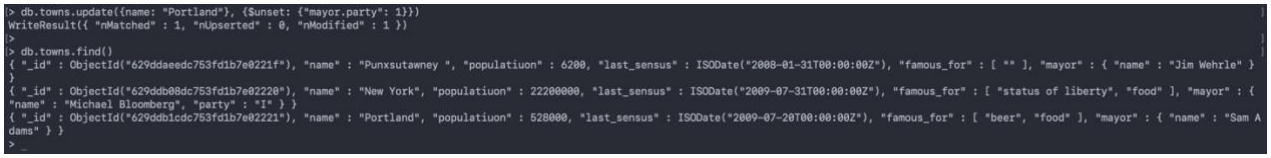
- Увеличим для всех самцов единорогов количество убитых вампиров на 5
  db.unicorns.update({gender: "m"}, {$inc: {vampires: 5}}, {multi: true})

```
[>
[> db.unicorns.update({gender: "m"}, {$inc: {vampires: 5}}, {multi: true})
WriteResult({ "nMatched" : 8, "nUpserted" : 0, "nModified" : 8 })
[>
[> db.unicorns.find()
{ "_id" : ObjectId("6297b4531a6efa3037002d7d"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" : 73 }
{ "_id" : ObjectId("6297b4531a6efa3037002d7e"), "name" : "Aurora", "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" : "f", "vampires" : 43 }
{ "_id" : ObjectId("6297b4531a6efa3037002d7f"), "name" : "Unicrom", "loves" : [ "energon", "redbull" ], "weight" : 984, "gender" : "m", "vampires" : 187 }
{ "_id" : ObjectId("6297b4531a6efa3037002d80"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 104 }
{ "_id" : ObjectId("6297b4531a6efa3037002d81"), "name" : "Solnara", "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("6297b4531a6efa3037002d82"), "name" : "Ayna", "weight" : 600, "vampires" : 51 }
{ "_id" : ObjectId("6297b4531a6efa3037002d83"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 44 }
{ "_id" : ObjectId("6297b4531a6efa3037002d84"), "name" : "Raleigh", "loves" : [ "rebull" ], "weight" : 421, "gender" : "m", "vampires" : 7 }
{ "_id" : ObjectId("6297b4531a6efa3037002d85"), "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("6297b4531a6efa3037002d86"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 59 }
{ "_id" : ObjectId("6297b4531a6efa3037002d87"), "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
{ "_id" : ObjectId("6297b5141a6efa3037002d88"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 170 }
{ "_id" : ObjectId("629e14c6dc753fd1b7e02222"), "name" : "Barny", "loves" : [ "grape" ], "weight" : 340, "gender" : "m", "vampires" : 5 }
> _
```

8.2.10

- Изменим информацию о городе Портланд: мэр этого города теперь беспартийный.

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



8.2.11

- Изменим информацию для единорога Pilot: теперь он любит и шоколад

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

8.2.12

- Изменим информацию о самке Aurora: теперь она любит еще и сахар, и лимоны

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

8.2.13

- Создадим коллекцию towns

```
[>
 > db.towns.insert({name: "Punxsutawney ",
 ... popujatiuon: 6200,
 ... last_sensus: ISODate("2008-01-31"),
 ... famous_for: ["phil the groundhog"],
 ... mayor: {
 ...     name: "Jim Wehrle"
 ... }}
[... )
 WriteResult({ "nInserted" : 1 })
[>
 > db.towns.insert({name: "New York",
 ... popujatiuon: 22200000,
 ... last_sensus: ISODate("2009-07-31"),
 ... famous_for: ["status of liberty", "food"],
 ... mayor: {
 ...     name: "Michael Bloomberg",
 ... party: "I"}}
[... )
 WriteResult({ "nInserted" : 1 })
[>
 > db.towns.insert({name: "Portland",
 ... popujatiuon: 528000,
 ... last_sensus: ISODate("2009-07-20"),
 ... famous_for: ["beer", "food"],
 ... mayor: {
 ...     name: "Sam Adams",
 ... party: "D"}}
[... )
 WriteResult({ "nInserted" : 1 })
[>
 > _
```

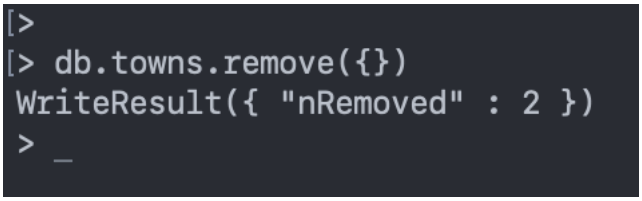
- Удалим элементы с беспартийными мэрами

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

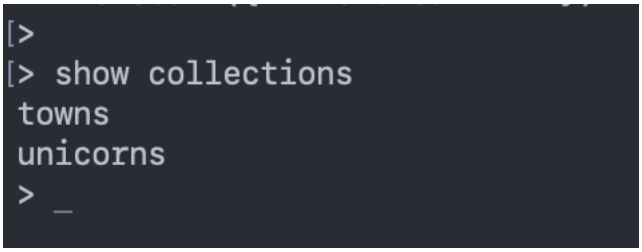
```
[> db.towns.count()
 3
[>
[> db.towns.remove({"mayor.party": {$exists: false}})
 WriteResult({ "nRemoved" : 1 })
[>
[> db.towns.count()
 2
[>
```

- Очистим коллекцию

  db.towns.remove({})



- Просмотрим список доступных коллекций



8.3.1

- Создадим коллекцию зон обитания единорогов

```
[>
[> db.habitats.insert({_id: "kr", name: "Kronverksky avenue", description: "ITMO building on Kronverksky avenue, 49"})
WriteResult({ "nInserted" : 1 })
[> db.habitats.insert({_id: "lm", name: "Lomonosova street", description: "ITMO building on Lomonosova street, 9"})
WriteResult({ "nInserted" : 1 })
[> db.habitats.insert({_id: "br", name: "Birzhevaya line", description: "ITMO building on Birzhevaya line, 16"})
WriteResult({ "nInserted" : 1 })
> _
```

- Включим для нескольких единорогов ссылку на место обитания, используя способ автоматического связывания

```
[>
[> db.unicorns.update({name: "Horny"}, {$set: {habitat: {$ref: "habitats", $id: "lm"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
[>
[> db.unicorns.update({name: "Pilot"}, {$set: {habitat: {$ref: "habitats", $id: "kr"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
[>
[> db.unicorns.update({name: "Roooooodles"}, {$set: {habitat: {$ref: "habitats", $id: "br"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> _
```

- Проверим только что обновленные документы:

```
[>
[> db.unicorns.find({habitat: {$exists: true}})
{ "_id" : ObjectId("6297b4531a6efa3037002d7d"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" : 73, "habitat" : DBRef("habitats", "lm") }
{ "_id" : ObjectId("6297b4531a6efa3037002d80"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 104, "habitat" : DBRef("habitats", "br") }
{ "_id" : ObjectId("6297b4531a6efa3037002d86"), "name" : "Pilot", "loves" : [ "apple", "watermelon", "chocolate" ], "weight" : 650, "gender" : "m", "vampires" : 59, "habitat" : DBRef("habitats", "kr") }
> _
```

8.3.2

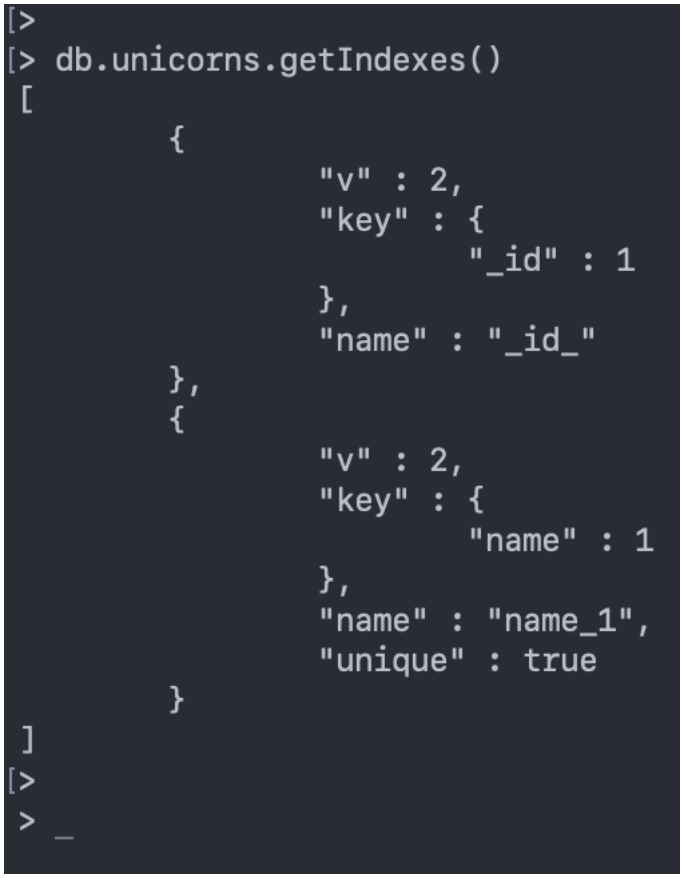
- Проверим, можно ли задать для коллекции *unicorns* индекс для ключа *name* с флагом *unique.*

```
[>
[> db.unicorns.createIndex({'name' : 1}, {'unique' : true})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
> _
```

Как видно, можно. Но мне не удалось сделать это с помощью команды *ensureIndex.*

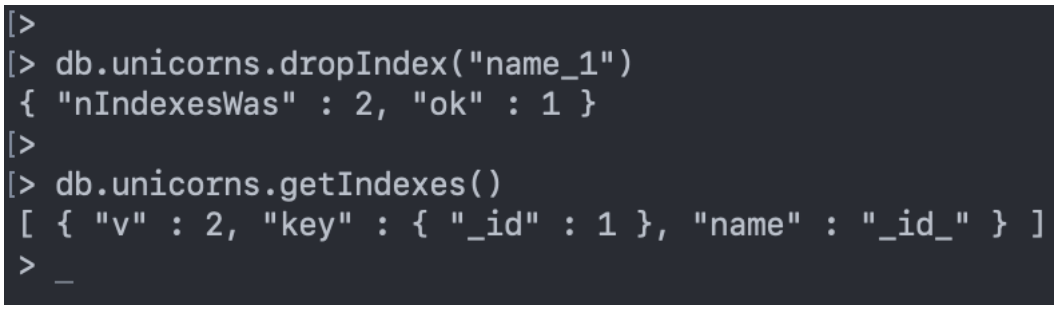
8.3.3

- Получим информацию о всех индексах коллекции *unicorns:* *db.unicorns.getIndexes()*



- Удалим все индексы, кроме индекса для идентификатора:

  db.unicorns.dropIndex("name_1")



- Попытаемся удалить индекс для идентификатора:
  db.unicorns.dropIndex("_id_")

```
[>
[> db.unicorns.dropIndex("_id_")
{
        "ok" : 0,
        "errmsg" : "cannot drop _id index",
        "code" : 72,
        "codeName" : "InvalidOptions"
}
> _
```

Ожидаемо получили ошибку *cannot drop _id index.*

8.3.4

- Создадим объемную коллекцию *numbers*, задействовав курсор:
  for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}

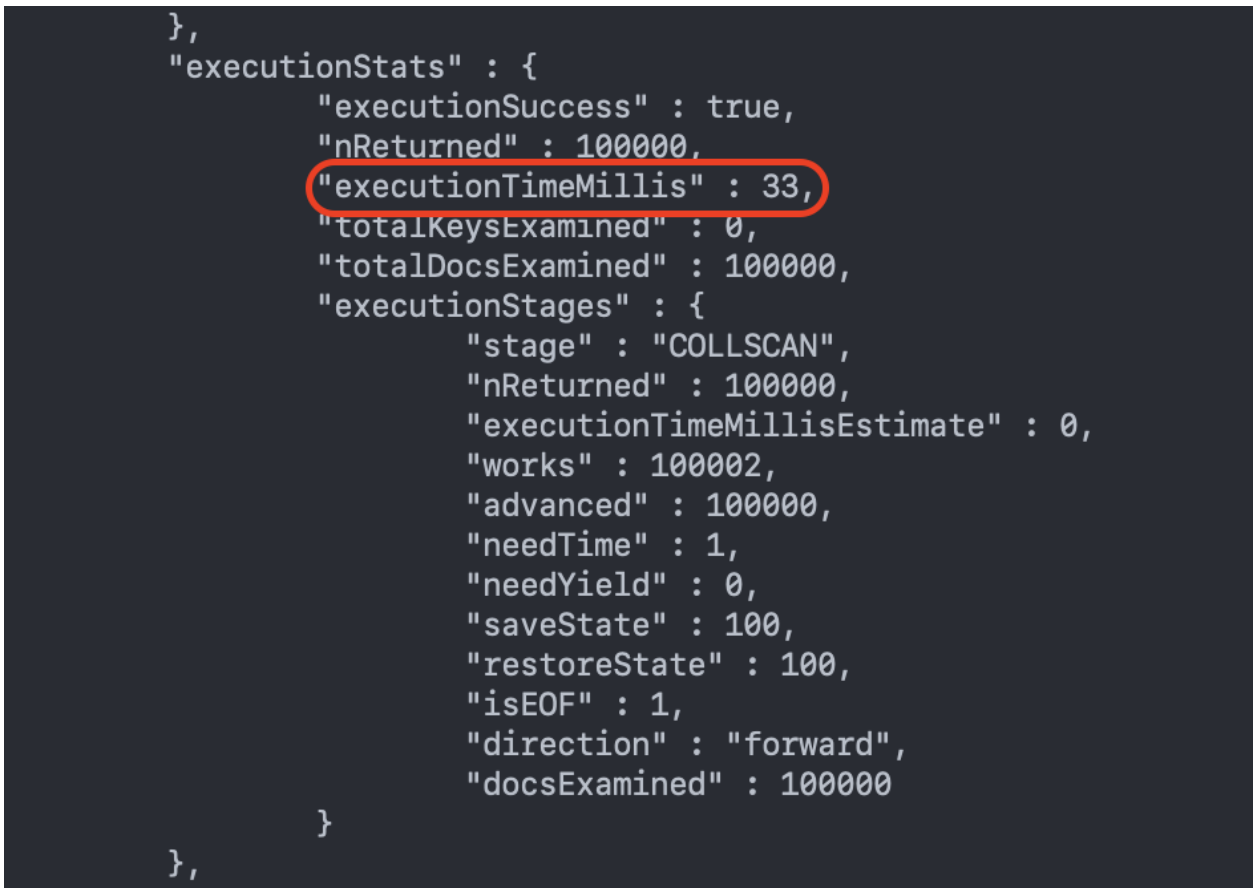
  Создание заняло около 30 секунд

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

```
[>
[> db.numbers.find().sort({ $natural: -1 }).limit(4)
{ "_id" : ObjectId("629e5c8cdc753fd1b7e1a8c5"), "value" : 99999 }
{ "_id" : ObjectId("629e5c8cdc753fd1b7e1a8c4"), "value" : 99998 }
{ "_id" : ObjectId("629e5c8cdc753fd1b7e1a8c3"), "value" : 99997 }
{ "_id" : ObjectId("629e5c8cdc753fd1b7e1a8c2"), "value" : 99996 }
> _
```

- Проанализируем план выполнения запроса, посмотрим, сколько потребовалось на него времени:

  db.numbers.explain("executionStats").find()



Видим, что запрос занял 33 миллисекунды

- Создадим индекс для ключа *value*, получим информацию о всех индексах

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

  db.numbers.getIndexes()

```
[>
[> db.numbers.createIndex({"value": 1})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
[>
[> db.numbers.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_"
        },
        {
                "v" : 2,
                "key" : {
                        "value" : 1
                },
                "name" : "value_1"
        }
]
> _
```

- Выполним запрос 2, проанализируем план выполнения запроса с установленным индексом:

```
[>
[> db.numbers.find().sort({ $natural: -1 }).limit(4)
{ "_id" : ObjectId("629e5c8cdc753fd1b7e1a8c5"), "value" : 99999 }
{ "_id" : ObjectId("629e5c8cdc753fd1b7e1a8c4"), "value" : 99998 }
{ "_id" : ObjectId("629e5c8cdc753fd1b7e1a8c3"), "value" : 99997 }
{ "_id" : ObjectId("629e5c8cdc753fd1b7e1a8c2"), "value" : 99996 }
[>
[> db.numbers.explain("executionStats").find()
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "learn.numbers",
                "indexFilterSet" : false,
                "parsedQuery" : {

                },
                "maxIndexedOrSolutionsReached" : false,
                "maxIndexedAndSolutionsReached" : false,
                "maxScansToExplodeReached" : false,
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 100000,
                "executionTimeMillis" : 34,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 100000,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "nReturned" : 100000,
                        "executionTimeMillisEstimate" : 1,
                        "works" : 100002,
                        "advanced" : 100000,
                        "needTime" : 1,
                        "needYield" : 0,
                        "saveState" : 100,
                        "restoreState" : 100,
                        "isEOF" : 1,
                        "direction" : "forward",
                        "docsExamined" : 100000
                }
        },
```

Заметим, что время выполнения запроса без индекса и время выполнения этого же запроса с индексом примерно одинаковы для данного случая.

ВЫВОДЫ

В процессе выполнения лабораторной работы, я познакомился с CRUD-операциями, вложенными объектами, агрегацией, ссылками и индексами в базе данных MongoDB.