

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

Мегафакультет информационных и трансляционных технологий

Проектирование и реализация баз данных

Лабораторная работа №5.2
"Работа с БД в СУБД MongoDB"

Работу выполнил:

Студент II курса
Коробковский В.А.

Группа:

К3242

Преподаватель:

Говорова М.М.

Санкт-Петербург
2022

Содержание

1. Цель работы	3
2. Практическое задание	3
3. Выполнение	3
3.1. Запрос №1	3
3.2. Запросы №2.1 и 2.2	4
3.3. Запрос №3	4
3.4. Запрос №4	5
3.5. Запрос №5	5
3.6. Запрос №6	5
3.7. Запрос №7	6
3.8. Запрос №8	6
3.9. Запрос №9	6
3.10. Запросы №10.1, №10.2 и №10.3	6
3.11. Запросы №11.1, №11.2 и №11.3	7
3.12. Запрос №12	8
3.13. Запрос №13	9
3.14. Запрос №14	10
3.15. Запросы №15.1 и №15.2	10
3.16. Запросы №16.1 и №16.2	11
3.17. Запросы №17.1 и №17.2	11
3.18. Запросы №18.1 и №18.2	12
3.19. Запросы №19.1 и №19.2	12
3.20. Запросы №20.1 и №20.2	13
3.21. Запросы №21.1 и №21.2	13
3.22. Запросы №22.1, №22.2, №22.3, №22.4 и №22.5	14
3.23. Запросы №23.1, №23.2 и №23.3	15
3.24. Запрос №24	17
3.25. Запросы №25.1, №25.2 и №25.3	17
3.26. Запросы №26.1, №26.2, №26.3, №26.4, №26.5, №26.6 и №26.7	18
4. Выводы	23

1. 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.

2. Практическое задание

Выполнить представленные запросы.

3. Выполнение

3.1. Запрос №1

Суть запроса:

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.

Результат выполнения запроса №1 можно посмотреть на рисунке 3.1, а код с полученным выводом — по этой [ссылке](#)

```
> 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});
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 })
> 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 })
> db.unicorns.find()
[{"_id": ObjectId("628a42455410ecae4ddee405"), "name": "Horny", "loves": ["carrot", "papaya"], "weight": 600, "gender": "m", "vampires": 63}, {"_id": ObjectId("628a42455410ecae4ddee406"), "name": "Aurora", "loves": ["carrot", "grape"], "weight": 450, "gender": "f", "vampires": 43}, {"_id": ObjectId("628a42455410ecae4ddee407"), "name": "Unicrom", "loves": ["energon", "redbull"], "weight": 984, "gender": "m", "vampires": 182}, {"_id": ObjectId("628a42455410ecae4ddee408"), "name": "Roooooodles", "loves": ["apple"], "weight": 575, "gender": "m", "vampires": 99}, {"_id": ObjectId("628a42455410ecae4ddee409"), "name": "Solnara", "loves": ["apple", "carrot", "chocolate"], "weight": 550, "gender": "f", "vampires": 80}, {"_id": ObjectId("628a42455410ecae4ddee40a"), "name": "Ayna", "loves": ["strawberry", "lemon"], "weight": 733, "gender": "f", "vampires": 40}, {"_id": ObjectId("628a42455410ecae4ddee40b"), "name": "Kenny", "loves": ["grape", "lemon"], "weight": 690, "gender": "m", "vampires": 39}, {"_id": ObjectId("628a42455410ecae4ddee40c"), "name": "Raleigh", "loves": ["apple", "sugar"], "weight": 421, "gender": "m", "vampires": 2}, {"_id": ObjectId("628a42455410ecae4ddee40d"), "name": "Leia", "loves": ["apple", "watermelon"], "weight": 601, "gender": "f", "vampires": 33}, {"_id": ObjectId("628a42455410ecae4ddee40e"), "name": "Pilot", "loves": ["apple", "watermelon"], "weight": 650, "gender": "m", "vampires": 54}, {"_id": ObjectId("628a42455410ecae4ddee40f"), "name": "Nimue", "loves": ["grape", "carrot"], "weight": 540, "gender": "f"}, {"_id": ObjectId("628a427c5410ecae4ddee410"), "name": "Dunx", "loves": ["grape", "watermelon"], "weight": 704, "gender": "m", "vampires": 165}]
```

Рисунок 3.1. — Результат выполнения запроса №1

3.2. Запросы №2.1 и 2.2

Суть запросов:

- Сформируйте запросы для вывода списков самцов и самок единорогов, ограничьте список самок первыми тремя особями и отсортируйте списки по имени.
- Найдите всех самок, которые любят carrot, ограничьте этот список первой особью с помощью функций findOne и limit.

Результат выполнения запроса №2.1 можно посмотреть на рисунке 3.2, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({gender: 'm'}).sort({name: 1})
{
  "_id": ObjectId("628a427c5410ecae4ddee410"),
  "name": "Dunx",
  "loves": [
    "grape",
    "watermelon"
  ],
  "weight": 704,
  "gender": "m",
  "vampires": 165
}
{
  "_id": ObjectId("628a42455410ecae4ddee405"),
  "name": "Horny",
  "loves": [
    "carrot",
    "papaya"
  ],
  "weight": 600,
  "gender": "m",
  "vampires": 63
}
{
  "_id": ObjectId("628a42455410ecae4ddee40b"),
  "name": "Kenny",
  "loves": [
    "grape",
    "lemon"
  ],
  "weight": 690,
  "gender": "m",
  "vampires": 39
}
{
  "_id": ObjectId("628a42455410ecae4ddee40e"),
  "name": "Pilot",
  "loves": [
    "apple",
    "watermelon"
  ],
  "weight": 650,
  "gender": "m",
  "vampires": 54
}
{
  "_id": ObjectId("628a42455410ecae4ddee40c"),
  "name": "Raleigh",
  "loves": [
    "apple",
    "sugar"
  ],
  "weight": 421,
  "gender": "m",
  "vampires": 2
}
{
  "_id": ObjectId("628a42455410ecae4ddee408"),
  "name": "Roooooodles",
  "loves": [
    "apple"
  ],
  "weight": 575,
  "gender": "m",
  "vampires": 99
}
{
  "_id": ObjectId("628a42455410ecae4ddee407"),
  "name": "Unicrom",
  "loves": [
    "energon",
    "redbull"
  ],
  "weight": 984,
  "gender": "m",
  "vampires": 182
}
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
{
  "_id": ObjectId("628a42455410ecae4ddee406"),
  "name": "Aurora",
  "loves": [
    "carrot",
    "grape"
  ],
  "weight": 450,
  "gender": "f",
  "vampires": 43
}
{
  "_id": ObjectId("628a42455410ecae4ddee40a"),
  "name": "Ayna",
  "loves": [
    "strawberry",
    "lemon"
  ],
  "weight": 733,
  "gender": "f",
  "vampires": 40
}
{
  "_id": ObjectId("628a42455410ecae4ddee40d"),
  "name": "Leia",
  "loves": [
    "apple",
    "watermelon"
  ],
  "weight": 601,
  "gender": "f",
  "vampires": 33
}
```

Рисунок 3.2. — Результат выполнения запроса №2.1

Результат выполнения запроса №2.2 можно посмотреть на рисунке 3.3, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  "_id": ObjectId("628a42455410ecae4ddee406"),
  "name": "Aurora",
  "loves": [
    "carrot",
    "grape"
  ],
  "weight": 450,
  "gender": "f",
  "vampires": 43
}
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
{
  "_id": ObjectId("628a42455410ecae4ddee406"),
  "name": "Aurora",
  "loves": [
    "carrot",
    "grape"
  ],
  "weight": 450,
  "gender": "f",
  "vampires": 43
}
```

Рисунок 3.3. — Результат выполнения запроса №2.2

3.3. Запрос №3

Суть запроса: Модифицируйте запрос для вывода списков самцов единорогов, исключив из результата информацию о предпочтениях и поле.

Результат выполнения запроса №3 можно посмотреть на рисунке 3.4, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({gender: "m"}, {loves: 0, gender: 0}).sort({name: 1})
{
  "_id": ObjectId("628a47945343b58a17f9703a"),
  "name": "Dunx",
  "weight": 704,
  "vampires": 165
}
{
  "_id": ObjectId("628a47935343b58a17f9702f"),
  "name": "Horny",
  "weight": 600,
  "vampires": 63
}
{
  "_id": ObjectId("628a47935343b58a17f97035"),
  "name": "Kenny",
  "weight": 690,
  "vampires": 39
}
{
  "_id": ObjectId("628a47945343b58a17f97038"),
  "name": "Pilot",
  "weight": 650,
  "vampires": 54
}
{
  "_id": ObjectId("628a47945343b58a17f97036"),
  "name": "Raleigh",
  "weight": 421,
  "vampires": 2
}
{
  "_id": ObjectId("628a47935343b58a17f97032"),
  "name": "Roooooodles",
  "weight": 575,
  "vampires": 99
}
{
  "_id": ObjectId("628a47935343b58a17f97031"),
  "name": "Unicrom",
  "weight": 984,
  "vampires": 182
}
```

Рисунок 3.4. — Результат выполнения запроса №3

3.4. Запрос №4

Суть запроса: Вывести список единорогов в обратном порядке добавления.

Результат выполнения запроса №4 можно посмотреть на рисунке 3.5, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find().sort({ $natural: -1 })
{ "_id": ObjectId("628a47945343b58a17f9703a"), "name": "Dunx", "loves": [ "grape", "watermelon" ], "weight": 704, "gender": "m", "vampires": 165 }
{ "_id": ObjectId("628a47945343b58a17f97039"), "name": "Nimue", "loves": [ "grape", "carrot" ], "weight": 540, "gender": "f" }
{ "_id": ObjectId("628a47945343b58a17f97038"), "name": "Pilot", "loves": [ "apple", "watermelon" ], "weight": 650, "gender": "m", "vampires": 54 }
{ "_id": ObjectId("628a47945343b58a17f97037"), "name": "Leia", "loves": [ "apple", "watermelon" ], "weight": 601, "gender": "f", "vampires": 33 }
{ "_id": ObjectId("628a47945343b58a17f97036"), "name": "Raleigh", "loves": [ "apple", "sugar" ], "weight": 421, "gender": "m", "vampires": 2 }
{ "_id": ObjectId("628a47935343b58a17f97035"), "name": "Kenny", "loves": [ "grape", "lemon" ], "weight": 690, "gender": "m", "vampires": 39 }
{ "_id": ObjectId("628a47935343b58a17f97034"), "name": "Ayna", "loves": [ "strawberry", "lemon" ], "weight": 733, "gender": "f", "vampires": 40 }
{ "_id": ObjectId("628a47935343b58a17f97033"), "name": "Roooooodles", "loves": [ "apple", "chocolate" ], "weight": 550, "gender": "f", "vampires": 80 }
{ "_id": ObjectId("628a47935343b58a17f97032"), "name": "Unicrom", "loves": [ "energon", "redbull" ], "weight": 984, "gender": "m", "vampires": 182 }
{ "_id": ObjectId("628a47935343b58a17f97031"), "name": "Aurora", "loves": [ "carrot", "grape" ], "weight": 450, "gender": "f", "vampires": 43 }
{ "_id": ObjectId("628a47935343b58a17f9702f"), "name": "Horny", "loves": [ "carrot", "papaya" ], "weight": 600, "gender": "m", "vampires": 63 }
```

Рисунок 3.5. — Результат выполнения запроса №4

3.5. Запрос №5

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

Результат выполнения запроса №5 можно посмотреть на рисунке 3.6, а код с полученным выводом — по этой [ссылке](#).

```
> 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, "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 }
>
```

Рисунок 3.6. — Результат выполнения запроса №5

3.6. Запрос №6

Суть запроса: Вывести список самок единорогов весом от 500 кг до 700 кг, исключив вывод идентификатора.

Результат выполнения запроса №6 можно посмотреть на рисунке 3.7, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({ gender: "f", weight: { $gt: 500, $lt: 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" }
```

Рисунок 3.7. — Результат выполнения запроса №6

3.7. Запрос №7

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

Результат выполнения запроса №7 можно посмотреть на рисунке 3.8, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({ gender: "m", weight: {$gt: 500}, loves: {$all: ["grape", "lemon"]}}, { _id: 0})
{ "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
>
```

Рисунок 3.8. — Результат выполнения запроса №7

3.8. Запрос №8

Суть запроса: Найти всех единорогов, не имеющих ключ vampires.

Результат выполнения запроса №8 можно посмотреть на рисунке 3.9, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({vampires: {$exists:false}})
{ "_id" : ObjectId("628b58a50f6391a3359d303b"), "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
>
```

Рисунок 3.9. — Результат выполнения запроса №8

3.9. Запрос №9

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

Результат выполнения запроса №9 можно посмотреть на рисунке 3.10, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({gender: "m"}, {loves: {$slice : 1}}).sort({name: 1})
{ "_id" : ObjectId("628b58ac0f6391a3359d303c"), "name" : "Dunx", "loves" : [ "grape" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
{ "_id" : ObjectId("628b58a40f6391a3359d3031"), "name" : "Horny", "loves" : [ "carrot" ], "weight" : 600, "gender" : "m", "vampires" : 63 }
{ "_id" : ObjectId("628b58a50f6391a3359d3037"), "name" : "Kenny", "loves" : [ "grape" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "_id" : ObjectId("628b58a50f6391a3359d303a"), "name" : "Pilot", "loves" : [ "apple" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("628b58a50f6391a3359d3038"), "name" : "Raleigh", "loves" : [ "apple" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "_id" : ObjectId("628b58a50f6391a3359d3034"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 99 }
{ "_id" : ObjectId("628b58a50f6391a3359d3033"), "name" : "Unicrom", "loves" : [ "energon" ], "weight" : 984, "gender" : "m", "vampires" : 182 }
>
```

Рисунок 3.10. — Результат выполнения запроса №9

3.10. Запросы №10.1, №10.2 и №10.3

Суть запросов:

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.

Результат выполнения запроса №10.1 можно посмотреть на рисунке 3.11, а код с полученным выводом — по этой [ссылке](#).

```
> db.towns.insert({name: "Punxsutawney ", populatioun: 6200, last_sensus: ISODate("2008-01-31"), famous_for: [], mayor: {name: "Jim Wehrle" }});  
WriteResult({ "nInserted" : 1 })  
> db.towns.insert({name: "New York", populatioun: 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", populatioun: 528000, last_sensus: ISODate("2009-07-28"), famous_for: ["beer", "food"], mayor: {name: "Sam Adams", party: "D"}});  
WriteResult({ "nInserted" : 1 })  
>
```

Рисунок 3.11. — Результат выполнения запроса №10.1

Результат выполнения запроса №10.2 можно посмотреть на рисунке 3.12, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.12. — Результат выполнения запроса №10.2

Результат выполнения запроса №10.3 можно посмотреть на рисунке 3.13, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.13. — Результат выполнения запроса №10.3

3.11. Запросы №11.1, №11.2 и №11.3

Суть запросов:

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.

Результат выполнения запроса №11.1 можно посмотреть на рисунке 3.14, а код с полученным выводом — по этой [ссылке](#).

```
> fn = function() {return this.gender == "m"}  
function() {return this.gender == "m"}  
> db.unicorns.find(fn)  
{ "_id" : ObjectId("628b58a40f6391a3359d3031"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" : 63 }  
{ "_id" : ObjectId("628b58a50f6391a3359d3033"), "name" : "Unicrom", "loves" : [ "energon", "redbull" ], "weight" : 984, "gender" : "m", "vampires" : 182 }  
{ "_id" : ObjectId("628b58a50f6391a3359d3034"), "name" : "Roooodoodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 99 }  
{ "_id" : ObjectId("628b58a50f6391a3359d3037"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }  
{ "_id" : ObjectId("628b58a50f6391a3359d3038"), "name" : "Raleigh", "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m", "vampires" : 2 }  
{ "_id" : ObjectId("628b58a50f6391a3359d303a"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 54 }  
{ "_id" : ObjectId("628b58a50f6391a3359d303c"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
```

Рисунок 3.14. — Результат выполнения запроса №11.1

Результат выполнения запроса №11.2 можно посмотреть на рисунке 3.15, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.15. — Результат выполнения запроса №11.2

Результат выполнения запроса №11.3 можно посмотреть на рисунке 3.16, а код с полученным выводом — по этой [ссылке](#).

```
> cursor.forEach(function(fn){print (fn.name);})  
Dunx  
Horny  
>
```

Рисунок 3.16. — Результат выполнения запроса №11.3

3.12. Запрос №12

Суть запроса: Вывести количество самок единорогов весом от 500 кг до 600 кг.

Результат выполнения запроса №12 можно посмотреть на рисунке 3.17, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.17. — Результат выполнения запроса №12

3.13. Запрос №13

Суть запроса: Вывести список предпочтений.

Результат выполнения запроса №13 можно посмотреть на рисунке 3.18, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.18. — Результат выполнения запроса №13

3.14. Запрос №14

Суть запроса: Посчитать количество особей единорогов обоих полов.

Результат выполнения запроса №14 можно посмотреть на рисунке 3.19, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.19. — Результат выполнения запроса №14

3.15. Запросы №15.1 и №15.2

Суть запросов:

1. Выполнить указанную в задании команду.
2. Проверить содержимое коллекции unicorns.

Результат выполнения запроса №15.1 можно посмотреть на рисунке 3.20, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.20. — Результат выполнения запроса №15.1

Результат выполнения запроса №15.2 можно посмотреть на рисунке 3.21, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find()
{ "_id" : ObjectId("628b58a40f6391a3359d3031"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" : 63 }
{ "_id" : ObjectId("628b58a50f6391a3359d3032"), "name" : "Aurora", "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" : "f", "vampires" : 43 }
{ "_id" : ObjectId("628b58a50f6391a3359d3033"), "name" : "Unicrom", "loves" : [ "energon", "redbull" ], "weight" : 984, "gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("628b58a50f6391a3359d3034"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 99 }
{ "_id" : ObjectId("628b58a50f6391a3359d3035"), "name" : "Solnara", "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("628b58a50f6391a3359d3036"), "name" : "Ayna", "loves" : [ "strawberry", "lemon" ], "weight" : 733, "gender" : "f", "vampires" : 40 }
{ "_id" : ObjectId("628b58a50f6391a3359d3037"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "_id" : ObjectId("628b58a50f6391a3359d3038"), "name" : "Raleigh", "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "_id" : ObjectId("628b58a50f6391a3359d3039"), "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 661, "gender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("628b58a50f6391a3359d303a"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("628b58a50f6391a3359d303b"), "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 548, "gender" : "f" }
{ "_id" : ObjectId("628b58a50f6391a3359d303c"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 764, "gender" : "m", "vampires" : 165 }
{ "_id" : ObjectId("628b63aa0f6391a3359d3040"), "name" : "Barny", "loves" : [ "grape" ], "weight" : 340, "gender" : "m" }
```

Рисунок 3.21. — Результат выполнения запроса №15.2

3.16. Запросы №16.1 и №16.2

Суть запросов:

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

Результат выполнения запроса №16.1 можно посмотреть на рисунке 3.22, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.update({name: "Ayna", gender: "f"}, {name: "Ayna", gender: "f", weight: 800, vampires: 51})  
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Рисунок 3.22. — Результат выполнения запроса №16.1

Результат выполнения запроса №16.2 можно посмотреть на рисунке 3.23, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({name: "Ayna"})  
{ "_id" : ObjectId("628b58a50f6391a3359d3036"), "name" : "Ayna", "gender" : "f", "weight" : 800, "vampires" : 51 }
```

Рисунок 3.23. — Результат выполнения запроса №16.2

3.17. Запросы №17.1 и №17.2

Суть запросов:

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

Результат выполнения запроса №17.1 можно посмотреть на рисунке 3.24, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.update({name: "Raleigh", gender: "m"}, {$set: {loves: "redbull"})}  
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Рисунок 3.24. — Результат выполнения запроса №17.1

Результат выполнения запроса №17.2 можно посмотреть на рисунке 3.25, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({name: "Raleigh"})  
{ "_id" : ObjectId("628b58a50f6391a3359d3038"), "name" : "Raleigh", "loves" : "redbull", "weight" : 421, "gender" : "m", "vampires" : 2 }
```

Рисунок 3.25. — Результат выполнения запроса №17.2

3.18. Запросы №18.1 и №18.2

Суть запросов:

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

Результат выполнения запроса №18.1 можно посмотреть на рисунке 3.26, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.update({gender: "m"}, {$inc: {vampires: 5}})  
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })  
>
```

Рисунок 3.26. — Результат выполнения запроса №18.1

Результат выполнения запроса №18.2 можно посмотреть на рисунке 3.27, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({gender: "m"})  
{ "_id" : ObjectId("628b58a40f6391a3359d3031"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" : 68 }  
{ "_id" : ObjectId("628b58a50f6391a3359d3033"), "name" : "Unicorn", "loves" : [ "energon", "redbull" ], "weight" : 984, "gender" : "m", "vampires" : 182 }  
{ "_id" : ObjectId("628b58a50f6391a3359d3034"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 99 }  
{ "_id" : ObjectId("628b58a50f6391a3359d3037"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }  
{ "_id" : ObjectId("628b58a50f6391a3359d3038"), "name" : "Raleigh", "loves" : [ "redbull" ], "weight" : 421, "gender" : "m", "vampires" : 2 }  
{ "_id" : ObjectId("628b58a50f6391a3359d303a"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 54 }  
{ "_id" : ObjectId("628b58ac0f6391a3359d303c"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 165 }  
{ "_id" : ObjectId("628b63aa0f6391a3359d3040"), "name" : "Barny", "loves" : [ "grape" ], "weight" : 340, "gender" : "m" }  
>
```

Рисунок 3.27. — Результат выполнения запроса №18.2

3.19. Запросы №19.1 и №19.2

Суть запросов:

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

Результат выполнения запроса №19.1 можно посмотреть на рисунке 3.28, а код с полученным выводом — по этой [ссылке](#).

```
> db.towns.update({name: "Portland"}, {$unset: {"mayor.party": 1}})  
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Рисунок 3.28. — Результат выполнения запроса №19.1

Результат выполнения запроса №19.2 можно посмотреть на рисунке 3.29, а код с полученным выводом — по этой [ссылке](#).

```
> db.towns.find()  
{ "_id" : ObjectId("628b5ca0f6391a3359d303d"), "name" : "Punxsutawney", "populatioun" : 6200, "last_sensus" : ISODate("2008-01-31T00:00:00Z"), "famous_for" : [ "" ], "mayor" : { "name" : "Jim Wehrle" } }  
{ "_id" : ObjectId("628b5ca0f6391a3359d303e"), "name" : "New York", "populatioun" : 22000000, "last_sensus" : ISODate("2009-07-31T00:00:00Z"), "famous_for" : [ "status of liberty", "food" ], "mayor" : { "name" : "Michael Bloomberg", "party" : "I" } }  
{ "_id" : ObjectId("628b5ca30f6391a3359d303f"), "name" : "Portland", "populatioun" : 520000, "last_sensus" : ISODate("2009-07-20T00:00:00Z"), "famous_for" : [ "beer", "food" ], "mayor" : { "name" : "Sam Adams" } }
```

Рисунок 3.29. — Результат выполнения запроса №19.2

3.20. Запросы №20.1 и №20.2

Суть запросов:

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

Результат выполнения запроса №20.1 можно посмотреть на рисунке 3.30, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.update({name : "Pilot", gender: "m"}, {$push: {loves: "chocolate"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Рисунок 3.30. — Результат выполнения запроса №20.1

Результат выполнения запроса №20.2 можно посмотреть на рисунке 3.31, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({name: "Pilot"})
{ "_id" : ObjectId("628b58a50f6391a3359d303a"), "name" : "Pilot", "loves" : [ "apple", "watermelon", "chocolate" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
>
```

Рисунок 3.31. — Результат выполнения запроса №20.2

3.21. Запросы №21.1 и №21.2

Суть запросов:

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

Результат выполнения запроса №21.1 можно посмотреть на рисунке 3.32, а код с полученным выводом — по этой [ссылке](#).

```
> db.users.update({name : "Aurora"}, {$addToSet: {loves: {$each: ["sugar", "lemon"]}}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

Рисунок 3.32. — Результат выполнения запроса №21.1

Результат выполнения запроса №21.2 можно посмотреть на рисунке 3.33, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find({name: "Aurora"})
{ "_id" : ObjectId("628b58a50f6391a3359d3032"), "name" : "Aurora", "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" : "f", "vampires" : 43 }
>
```

Рисунок 3.33. — Результат выполнения запроса №21.2

3.22. Запросы №22.1, №22.2, №22.3, №22.4 и №22.5

Суть запросов:

1. Создайте коллекцию towns, включающую документы из задания.
2. Удалите документы с беспартийными мэрами.
3. Проверить содержимое коллекции towns.
4. Очистите коллекцию towns.
5. Просмотрите список доступных коллекций.

Результат выполнения запроса №22.1 можно посмотреть на рисунке 3.34, а код с полученным выводом — по этой [ссылке](#).

```
> db.towns.insert({name: "Punxsutawney ", popujatiou: 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", popujatiou: 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", popujatiou: 528000, last_sensus: ISODate("2009-07-28"), famous_for: ["beer", "food"], mayor: {name: "Sam Adams", party: "D"}})
WriteResult({ "nInserted" : 1 })
```

Рисунок 3.34. — Результат выполнения запроса №22.1

Результат выполнения запроса №22.2 можно посмотреть на рисунке 3.35, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.35. — Результат выполнения запроса №22.2

Результат выполнения запроса №22.3 можно посмотреть на рисунке 3.36, а код с полученным выводом — по этой [ссылке](#).

```
> db.towns.find()
[{"_id": ObjectId("628094c30f6391a3359d3042"), "name": "New York", "popujatiou": 22200000, "last_sensus": ISODate("2009-07-31T00:00:00Z"), "famous_for": ["status of liberty", "food"], "mayor": { "name": "Michael Bloomberg", "party": "I" } },
 {"_id": ObjectId("628094c40f6391a3359d3043"), "name": "Portland", "popujatiou": 528000, "last_sensus": ISODate("2009-07-28T00:00:00Z"), "famous_for": ["beer", "food"], "mayor": { "name": "Sam Adams", "party": "D" } }
```

Рисунок 3.36. — Результат выполнения запроса №22.3

Результат выполнения запроса №22.4 можно посмотреть на рисунке 3.37, а код с полученным выводом — по этой [ссылке](#).

```
> db.towns.remove({})
WriteResult({ "nRemoved" : 2 })
>
```

Рисунок 3.37. — Результат выполнения запроса №22.4

Результат выполнения запроса №22.5 можно посмотреть на рисунке 3.38, а код с полученным выводом — по этой [ссылке](#).

```
> show collections
towns
unicorns
> db.towns.find()
>
```

Рисунок 3.38. — Результат выполнения запроса №22.5

3.23. Запросы №23.1, №23.2 и №23.3

Суть запросов:

1. Создайте коллекцию towns, включающую документы из задания.
2. Удалите документы с беспартийными мэрами.
3. Проверить содержимое коллекции towns.

Результат выполнения запроса №23.1 можно посмотреть на рисунке 3.39, а код с полученным выводом — по этой [ссылке](#).

```
> db.areas.insert({_id: "Bar", short: "Bar", full: "Barcelona", description: "Libertad para Cataluña!"})
WriteResult({ "nInserted" : 1 })
> db.areas.insert({_id: "Liv", short: "Liv", full: "Liverpool", description: "You will never walk alone!"})
WriteResult({ "nInserted" : 1 })
> db.areas.insert({_id: "Tur", short: "Tur", full: "Turin", description: "Storia di un grande amore!"})
WriteResult({ "nInserted" : 1 })
> db.areas.insert({_id: "Spb", short: "Spb", full: "Saint-Petersburg", description: "Город над вольной Невой. Идёт волна!"})
WriteResult({ "nInserted" : 1 })
> db.areas.insert({_id: "Par", short: "Par", full: "Paris", description: "Ici c'est Paris! C'est la ville de l'amour!"})
WriteResult({ "nInserted" : 1 })
> db.areas.insert({_id: "Mun", short: "Mun", full: "Munich", description: "Mia San Mia!"})
WriteResult({ "nInserted" : 1 })
> db.areas.find()
[{"_id": "Bar", "short": "Bar", "full": "Barcelona", "description": "Libertad para Cataluña!" },
 {"_id": "Liv", "short": "Liv", "full": "Liverpool", "description": "You will never walk alone!" },
 {"_id": "Tur", "short": "Tur", "full": "Turin", "description": "Storia di un grande amore!" },
 {"_id": "Spb", "short": "Spb", "full": "Saint-Petersburg", "description": "Город над вольной Невой. Идёт волна!" },
 {"_id": "Par", "short": "Par", "full": "Paris", "description": "Ici c'est Paris! C'est la ville de l'amour!" },
 {"_id": "Mun", "short": "Mun", "full": "Munich", "description": "Mia San Mia!" }]
```

Рисунок 3.39. — Результат выполнения запроса №23.1

Результат выполнения запроса №23.2 можно посмотреть на рисунке 3.40, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.update({name: "Horny"}, {$set: {"city": {$ref:"areas", $id: "Bar"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Aurora"}, {$set: {"city": {$ref:"areas", $id: "Bar"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Unicrom"}, {$set: {"city": {$ref:"areas", $id: "Liv"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Roooooodles"}, {$set: {"city": {$ref:"areas", $id: "Liv"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Solnara"}, {$set: {"city": {$ref:"areas", $id: "Tur"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Ayna"}, {$set: {"city": {$ref:"areas", $id: "Tur"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Kenny"}, {$set: {"city": {$ref:"areas", $id: "Spb"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Raleigh"}, {$set: {"city": {$ref:"areas", $id: "Spb"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Leia"}, {$set: {"city": {$ref:"areas", $id: "Par"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Pilot"}, {$set: {"city": {$ref:"areas", $id: "Par"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Nimue"}, {$set: {"city": {$ref:"areas", $id: "Mun"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Dunx"}, {$set: {"city": {$ref:"areas", $id: "Mun"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Рисунок 3.40. — Результат выполнения запроса №23.2

Результат выполнения запроса №23.3 можно посмотреть на рисунке 3.41, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.find()
[{"_id": ObjectId("628b9ade0f6391a3359d305b"), "name": "Horny", "loves": ["carrot", "papaya"], "weight": 600, "gender": "m", "vampires": 63, "city": DBRef("areas", "Bar") },
 {"_id": ObjectId("628b9ade0f6391a3359d305c"), "name": "Aurora", "loves": ["carrot", "grape"], "weight": 450, "gender": "f", "vampires": 43, "city": DBRef("areas", "Bar") },
 {"_id": ObjectId("628b9ade0f6391a3359d305d"), "name": "Unicrom", "loves": ["ergon", "redbull"], "weight": 984, "gender": "m", "vampires": 182, "city": DBRef("areas", "Liv") },
 {"_id": ObjectId("628b9ade0f6391a3359d305e"), "name": "Rooooodles", "loves": ["apple"], "weight": 575, "gender": "m", "vampires": 99, "city": DBRef("areas", "Liv") },
 {"_id": ObjectId("628b9ade0f6391a3359d305f"), "name": "Solnara", "loves": ["apple", "carrot", "chocolate"], "weight": 550, "gender": "f", "vampires": 80, "city": DBRef("areas", "Tur") },
 {"_id": ObjectId("628b9ade0f6391a3359d3060"), "name": "Ayna", "loves": ["strawberry", "lemon"], "weight": 733, "gender": "f", "vampires": 40, "city": DBRef("areas", "Tur") },
 {"_id": ObjectId("628b9ade0f6391a3359d3061"), "name": "Kenny", "loves": ["grape", "lemon"], "weight": 690, "gender": "m", "vampires": 39, "city": DBRef("areas", "Spb") },
 {"_id": ObjectId("628b9ade0f6391a3359d3062"), "name": "Raleigh", "loves": ["apple", "sugar"], "weight": 421, "gender": "m", "vampires": 2, "city": DBRef("areas", "Spb") },
 {"_id": ObjectId("628b9ade0f6391a3359d3063"), "name": "Leia", "loves": ["apple", "watermelon"], "weight": 601, "gender": "f", "vampires": 33, "city": DBRef("areas", "Par") },
 {"_id": ObjectId("628b9ade0f6391a3359d3064"), "name": "Pilot", "loves": ["apple", "watermelon"], "weight": 650, "gender": "m", "vampires": 54, "city": DBRef("areas", "Par") },
 {"_id": ObjectId("628b9ade0f6391a3359d3065"), "name": "Nimue", "loves": ["grape", "carrot"], "weight": 540, "gender": "f", "city": DBRef("areas", "Mun") },
 {"_id": ObjectId("628b9ad0f0f6391a3359d3066"), "name": "Dunx", "loves": ["grape", "watermelon"], "weight": 784, "gender": "m", "vampires": 165, "city": DBRef("areas", "Mun") }]
```

Рисунок 3.41. — Результат выполнения запроса №23.3

3.24. Запрос №24

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

Результат выполнения запроса №24 можно посмотреть на рисунке 3.42, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.insert({name: 'Horny', dob: new Date(1992,2,13,7,47), loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Aurora', dob: new Date(1991, 0, 24, 13, 0), loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Unicrom', dob: new Date(1973, 1, 9, 22, 10), loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Roooooodles', dob: new Date(1979, 7, 18, 18, 44), loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Sölnara', dob: new Date(1985, 6, 4, 2, 1), loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Ayna', dob: new Date(1998, 2, 7, 8, 30), loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Kenny', dob: new Date(1997, 6, 1, 10, 42), loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Raleigh', dob: new Date(2005, 4, 3, 0, 57), loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Leia', dob: new Date(2001, 9, 8, 14, 53), loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Pilot', dob: new Date(1997, 2, 1, 5, 3), loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Níne', dob: new Date(1999, 11, 20, 16, 15), loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
WriteResult({ "nInserted" : 1 })
> db.unicorns.insert({name: 'Dunx', dob: new Date(1976, 6, 18, 18, 18), loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165});
WriteResult({ "nInserted" : 1 })
> db.unicorns.ensureIndex({"name" : 1}, {"unique" : true})
uncaught exception: TypeError: db.unicorns.ensureIndex is not a function
@(shell):1:1
>
```

Рисунок 3.42. — Результат выполнения запроса №24

3.25. Запросы №25.1, №25.2 и №25.3

Суть запросов:

1. Создайте коллекцию towns, включающую документы из задания.
2. Удалите документы с беспартийными мэрами.
3. Проверить содержимое коллекции towns.

Результат выполнения запроса №25.1 можно посмотреть на рисунке 3.43, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.43. — Результат выполнения запроса №25.1

Результат выполнения запроса №25.2 можно посмотреть на рисунке 3.44, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.dropIndexes()
{
    "nIndexesWas" : 1,
    "msg" : "non-_id indexes dropped for collection",
    "ok" : 1
}
>
```

Рисунок 3.44. — Результат выполнения запроса №25.2

Результат выполнения запроса №25.3 можно посмотреть на рисунке 3.45, а код с полученным выводом — по этой [ссылке](#).

```
> db.unicorns.dropIndex({"_id": 1})
{
    "ok" : 0,
    "errmsg" : "cannot drop _id index",
    "code" : 72,
    "codeName" : "InvalidOptions"
}
>
```

Рисунок 3.45. — Результат выполнения запроса №25.3

3.26. Запросы №26.1, №26.2, №26.3, №26.4, №26.5, №26.6 и №26.7

Суть запросов:

1. Создайте объемную коллекцию numbers, задействовав курсор.
2. Выберите четыре последних документа.
3. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса?
4. Создайте индекс для ключа value.
5. Получите информацию о всех индексах коллекции numbers.
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?

Результат выполнения запроса №26.1 можно посмотреть на рисунке 3.46, а код с полученным выводом — по этой [ссылке](#).

```
> db.createCollection("numbers")
{ "ok" : 1 }
> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
WriteResult({ "nInserted" : 1 })
>
```

Рисунок 3.46. — Результат выполнения запроса №26.1

Результат выполнения запроса №26.2 можно посмотреть на рисунке 3.47, а код с полученным выводом — по этой [ссылке](#).

```
> db.numbers.find().sort({ $natural: -1 }).limit(4)
{ "_id" : ObjectId("628ba0070f6391a3359eb712"), "value" : 99999 }
{ "_id" : ObjectId("628ba0070f6391a3359eb711"), "value" : 99998 }
{ "_id" : ObjectId("628ba0070f6391a3359eb710"), "value" : 99997 }
{ "_id" : ObjectId("628ba0070f6391a3359eb70f"), "value" : 99996 }
>
```

Рисунок 3.47. — Результат выполнения запроса №26.2

Результат выполнения запроса №26.3 можно посмотреть на рисунке 3.48 (последняя строчка является ответом на вопрос), а код с полученным выводом — по этой [ссылке](#).

```
> db.numbers.explain("executionStats").find().sort({ $natural: -1 }).limit(4)
{
  "explainVersion" : "1",
  "queryPlanner" : {
    "namespace" : "learn.numbers",
    "indexFilterSet" : false,
    "parsedQuery" : {

    },
    "maxIndexedOrSolutionsReached" : false,
    "maxIndexedAndSolutionsReached" : false,
    "maxScansToExplodeReached" : false,
    "winningPlan" : {
      "stage" : "LIMIT",
      "limitAmount" : 4,
      "inputStage" : {
        "stage" : "COLLSCAN",
        "direction" : "backward"
      }
    },
    "rejectedPlans" : [ ]
  },
  "executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 4,
    "executionTimeMillis" : 2,
```

Рисунок 3.48. — Результат выполнения запроса №26.3

Результат выполнения запроса №26.4 можно посмотреть на рисунке 3.49, а код с полученным выводом — по этой [ссылке](#).

```
> db.numbers.createIndex({"value" : 1})
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
>
```

Рисунок 3.49. — Результат выполнения запроса №26.4

Результат выполнения запроса №26.5 можно посмотреть на рисунке 3.50, а код с полученным выводом — по этой [ссылке](#).

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

Рисунок 3.50. — Результат выполнения запроса №26.5

Результат выполнения запроса №26.6 можно посмотреть на рисунке 3.51, а код с полученным выводом — по этой [ссылке](#).

```
> db.numbers.find().sort({ $natural: -1 }).limit(4)
{ "_id" : ObjectId("628ba0070f6391a3359eb712"), "value" : 99999 }
{ "_id" : ObjectId("628ba0070f6391a3359eb711"), "value" : 99998 }
{ "_id" : ObjectId("628ba0070f6391a3359eb710"), "value" : 99997 }
{ "_id" : ObjectId("628ba0070f6391a3359eb70f"), "value" : 99996 }
>
```

Рисунок 3.51. — Результат выполнения запроса №26.6

Результат выполнения запроса №26.7 можно посмотреть на рисунке 3.52 (последняя строчка является ответом на вопрос), а код с полученным выводом — по этой [ссылке](#).

```
> db.numbers.explain("executionStats").find().sort({ $natural: -1 }).limit(4)
{
  "explainVersion" : "1",
  "queryPlanner" : {
    "namespace" : "learn.numbers",
    "indexFilterSet" : false,
    "parsedQuery" : {

    },
    "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" : 0
  }
}
```

Рисунок 3.52. — Результат выполнения запроса №26.7

Скорости почти равны, однако с индексом получилось чуть быстрее. Это означает, что данных всё ещё мало, однако по сравнению с запросами с и без индексов в З лабораторной работе, тут скорость с индексом уже меньше скорости без индекса, то есть теперь можно сделать точный вывод о том, что скорость выполнения запроса с индексом уменьшается с увеличением количества данных.

4. Выводы

Во время выполнения лабораторной работы я познакомился с системой управления БД под названием MongoDB; научился добавлять, обновлять, удалять данные из таблиц; создавать коллекции, индексы; писать простейшие запросы.