Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»
Факультет инфокоммуникационных технологий

Лабораторная работа №8

«Работа с БД в СУБД MongoDB»

по дисциплине

«Проектирование и реализация баз данных»

Выполнил:

студент II курса ФИКТ

группы К3241

Ф.И.О. Кондрашов Егор Юрьевич

Проверила:

Говорова Марина Михайловна

Санкт-Петербург

2021

Цель лабораторной 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 практического задания:

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

```
> db.unicorns.find({gender: "m"}).limit(3).sort({name: 1})
{ "_id" : ObjectId("60b51d7a870280cf85079b69"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "
{ "_id" : ObjectId("60b51b84870280cf85079b5e"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "we
{ "_id" : ObjectId("60b51cca870280cf85079b64"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weig
```

```
> db.unicorns.find({gender: "f"}).limit(3).sort({name: 1})
{ "_id" : ObjectId("60b51bac870280cf85079b5f"), "name" : "Aurora", "loves" : [ "carrot"
{ "_id" : ObjectId("60b51ca7870280cf85079b63"), "name" : "Ayna", "loves" : [ "strawberr
{ "_id" : ObjectId("60b51d12870280cf85079b66"), "name" : "Leia", "loves" : [ "apple", "
>
```

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

```
> db.unicorns.find({gender: "f", loves: "carrot"}).limit(1)
{ "_id" : ObjectId("60b51bac870280cf85079b5f"), "name" : "Aurora", "loves" : [ "carrot", "grape" ], "
```

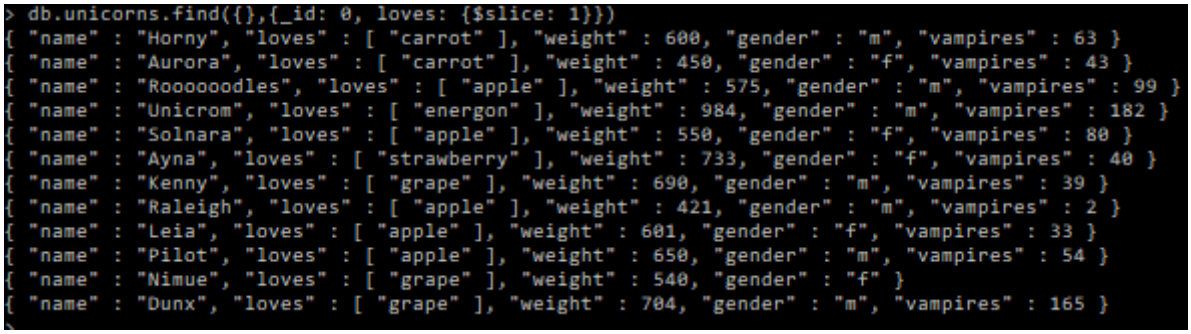
```
> db.unicorns.findOne({gender: "f", loves: "carrot"})
{
        "_id" : ObjectId("60b51bac870280cf85079b5f"),
        "name" : "Aurora",
        "loves" : [
                "carrot",
                "grape"
        ],
        "weight" : 450,
        "gender" : "f",
        "vampires" : 43
}
```

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

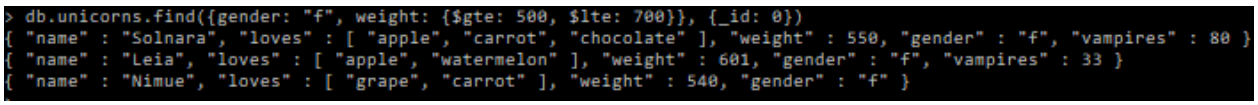
```
> db.unicorns.find({gender: "m"}, {gender: 0, loves: 0}).limit(3).sort({name: 1})
{ "_id" : ObjectId("60b51d7a870280cf85079b69"), "name" : "Dunx", "weight" : 704, "vampires" : 165 }
{ "_id" : ObjectId("60b51b84870280cf85079b5e"), "name" : "Horny", "weight" : 600, "vampires" : 63 }
{ "_id" : ObjectId("60b51cca870280cf85079b64"), "name" : "Kenny", "weight" : 690, "vampires" : 39 }
>
```

8.1.4 Вывести список единорогов в обратном порядке добавления.

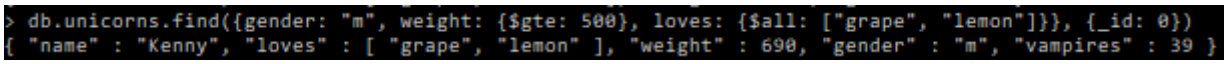
8.1.5 Вывести список единорогов с названием первого любимого предпочтения, исключив идентификатор.



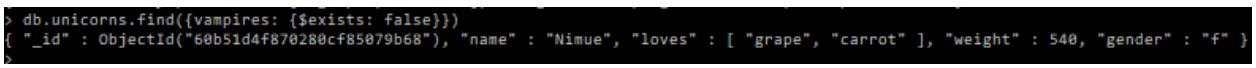
8.1.6 Вывести список самок единорогов весом от полутонны до 700 кг, исключив вывод идентификатора.



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



8.1.8 Найти всех единорогов, не имеющих ключ vampires.



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

```
> db.unicorns.find({gender: "m"},{name: 1, _id: 0, loves: {$slice: 1}}).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 2) Сформировать запрос, который возвращает список городов с независимыми мэрами (party=”I”). Вывести только название города и информацию о мэре.

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

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

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

8.2.2 1) Сформировать функцию для вывода списка самцов единорогов.

```
> fn = function () {return this.gender=="m";}
```

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

```
> var cursor = db.unicorns.find(fn);null;
null
> cursor.limit(2).sort({name: 1})
{ "_id" : ObjectId("60b51d7a870280cf85079b69"), "name" : "Dunx", "loves" : [ "
ender" : "m", "vampires" : 165 }
{ "_id" : ObjectId("60b51b84870280cf85079b5e"), "name" : "Horny", "loves" : [
der" : "m", "vampires" : 63 }
```

3) Вывести результат, используя forEach.

```
> var cursor = db.unicorns.find(fn).limit(2).sort({name: 1})
> cursor.forEach(function(obj) { print (obj.name); })
Dunx
Horny
```

8.2.3 Вывести количество самок единорогов весом от полутонны до 600 кг.

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

8.2.4 Вывести список предпочтений.

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

8.2.5 Посчитать количество особей единорогов обоих полов.

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

8.2.7 Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вампира.

```
> db.unicorns.update({name: "Ayna"}, {name: "Ayna", loves: ["strawberry", "lemon"],gender: "f", weight: 800, vampires: 5
1})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

8.2.8 Для самца единорога Raleigh внести изменения в БД: теперь он любит рэдбул.

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

8.2.9 Всем самцам единорогов увеличить количество убитых вапмиров на 5.

```
> db.unicorns.update({gender: "m"}, {$inc: {vampires: 5}}, {multi: true})
WriteResult({ "nMatched" : 8, "nUpserted" : 0, "nModified" : 8 })
```

8.2.10 Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.

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

```
> db.towns.find()
{ "_id" : ObjectId("60b5c49d3d831a30e2159309"), "name" : "Punxsutawney", "population" : 6200,
008-01-30T00:00:00Z"), "famous_for" : [ "" ], "mayor" : { "name" : "Jim Wehrle" } }
{ "_id" : ObjectId("60b5c5013d831a30e215930a"), "name" : "New York", "population" : 22200000,
009-07-31T00:00:00Z"), "famous_for" : [ "statue of liberty", "food" ], "mayor" : { "name" : "M
 : "I" } }
{ "_id" : ObjectId("60b5c5353d831a30e215930b"), "name" : "Portland", "population" : 528000, "l
9-07-20T00:00:00Z"), "famous_for" : [ "beer", "food" ], "mayor" : { "name" : "Sam Adams" } }
```

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

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

```
> db.unicorns.find({name: "Pilot"})
{ "_id" : ObjectId("60b51d2a870280cf85079b67"), "name" : "Pilot", "loves" : [ "apple", "watermelon", "chocolate" ],
ght" : 650, "gender" : "m", "vampires" : 59 }
```

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

```
> db.unicorns.update({name: "Aurora"}, {$addToSet: {loves: {$each: ["sugar", "lemon"]}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

8.2.13 Удалите документы с беспартийными мэрами. Проверьте содержание коллекции. Очистите коллекцию. Просмотрите список доступных коллекций.

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

```
> db.towns.find()
{ "_id" : ObjectId("60b5c5013d831a30e215930a"), "name" : "New York",
mayor" : { "name" : "Michael Bloomberg", "party" : "I" } }
{ "_id" : ObjectId("60b5c5353d831a30e215930b"), "name" : "Portland",
e" : "Sam Adams", "party" : "D" } }
```

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

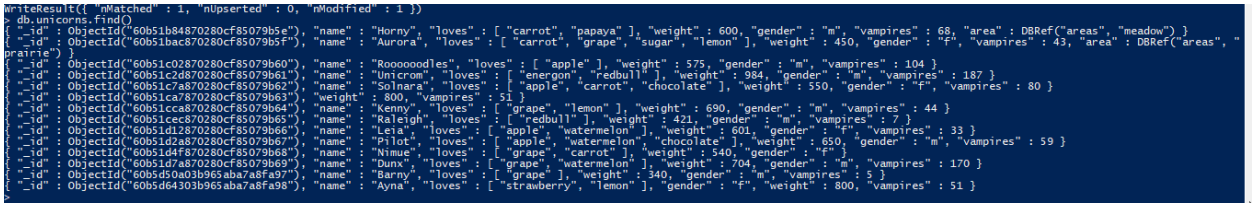
```
> show collections
towns
unicorns
> db.towns.find()
>
```

8.3.1

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

```
> db.areas.insert({_id: "meadow", name: "Green Meadow", desc: "Situated in Russia"})
WriteResult({ "nInserted" : 1 })
> db.areas.insert({_id: "prairie", name: "Blue Prairie", desc: "Situated in USA"})
WriteResult({ "nInserted" : 1 })
```

```
> db.unicorns.update({name: "Horny"}, {$set: {area: {$ref: "areas", $id: "meadow"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.update({name: "Aurora"}, {$set: {area: {$ref: "areas", $id: "prairie"}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```




```
> db.areas.find({_id: db.unicorns.findOne({name: "Horny"}).area.$id})
{ "_id" : "meadow", "name" : "Green Meadow", "desc" : "Situated in Russia" }
>
```

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

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

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

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

```
> db.unicorns.dropIndex("name_1")
{ "nIndexesWas" : 2, "ok" : 1 }
>
```

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

8.3.4 1) Создайте объемную коллекцию numbers, задействовав курсор. 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?

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

```
> db.numbers.explain("executionStats").find({value: {$gte: 99996}})
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "learn.numbers",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "value" : {
                                "$gte" : 99996
                        }
                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "value" : 1
                                },
                                "indexName" : "value_1",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                        "value" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "value" : [
                                                "[99996.0, inf.0]"
                                        ]
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 4,
                "executionTimeMillis" : 29,
                "totalKeysExamined" : 4,
                "totalDocsExamined" : 4,
                "executionStages" : {
                        "stage" : "FETCH",
                        "nReturned" : 4,
                        "executionTimeMillisEstimate" : 10,
                        "works" : 5,
                        "advanced" : 4,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
                        "isEOF" : 1,
                        "docsExamined" : 4,
                        "alreadyHasObj" : 0,
```

Как видно, при создании индекса на поле “value”, скорость выполнения запроса увеличилась с 48 до 29 мс, то есть на 40%. Таким образом, очевидно, что для ускорения часто применяемых запросов стоит создавать индексы.

Вывод:

В ходе выполнения работы были созданы запросы на вставку, изменение, удаление документов в MongoDB, на выборку данных по критериям, на работу с индексами и ссылками.