Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №6 «Работа с БД в СУБД MongoDB»

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Зеленин Д.С.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 6.0.6 (текущая).

Практическое задание 2.1.1:

Создаем бд learn

```
> use learn
< switched to db learn
```

Заполняем данные коллекции unicorns

1 способ

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

2 способ

```
> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
< {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
```

Проверяем данные

```
db.unicorns.find()
{
  _id: ObjectId('6596a3f09dbc95074933cbe7'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId('6596a3f09dbc95074933cbe8'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId('6596a3f09dbc95074933cbe9'),
  name: 'Unicrom',
```

Практическое задание 2.2.1

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

Для самцов:

```
> db.unicorns.find({gender:'m'}).sort({name:1})
< {
    _id: ObjectId('6596a3f09dbc95074933cbe7'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('6596a3f09dbc95074933cbed'),
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
  {
    _id: ObjectId('6596a3f09dbc95074933cbf0'),
```

Для самок:

```
> db.unicorns.find({gender: 'f'}).sort({name:1}).limit(3)
< {
    _id: ObjectId('6596a3f09dbc95074933cbe8'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('6596a3f09dbc95074933cbec'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
```

```
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
    _id: ObjectId('6596a3f09dbc95074933cbef'),
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
learn>
```

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

1 способ

```
> db.unicorns.findOne({loves : "carrot", gender : "f"})
< {
    _id: ObjectId('6596a3f09dbc95074933cbe8'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
learn>
```

2 способ

```
> db.unicorns.find({loves : "carrot", gender : "f"}).limit(1)
< {
    _id: ObjectId('6596a3f09dbc95074933cbe8'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

Практическое задание 2.2.2

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

```
> db.unicorns.find({gender : 'm'}, {loves : 0, gender : 0})
< {
    _id: ObjectId('6596a3f09dbc95074933cbe7'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
  {
    _id: ObjectId('6596a3f09dbc95074933cbe9'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
  {
    _id: ObjectId('6596a3f09dbc95074933cbea'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  }
  {
```

Практическое задание 2.2.3

Вывести список единорогов в обратном порядке добавления

```
> db.unicorns.find().sort({$natural : -1})
< {
    _id: ObjectId('6596a3f09dbc95074933cbf1'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
  {
    _id: ObjectId('6596a3f09dbc95074933cbf0'),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
  {
    _id: ObjectId('6596a3f09dbc95074933cbef'),
    name: 'Leia',
```

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.

```
> 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'
    ],
```

Практическое задание 2.3.2

Вывести 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 : 0})
< {
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
```

Практическое задание 2.3.3

Найти всех единорогов, не имеющих ключ `vampires`.

```
> db.unicorns.find({vampires : {$exists : false}})
< {
    _id: ObjectId('6596a3f09dbc95074933cbf1'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
learn>
```

Практическое задание 2.3.4

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

```
> db.unicorns.find({gender : 'm'}, {loves : { $slice : 1}}).sort({name : 1})
< {
    _id: ObjectId('6596a3f09dbc95074933cbe7'),
    name: 'Horny',
    loves: [
      'carrot'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('6596a3f09dbc95074933cbed'),
    name: 'Kenny',
    loves: [
      'grape'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
```

Практическое задание 3.1.1

Создайте коллекцию towns, включающую следующие документы:

```
db.towns.find()
{
  _id: ObjectId('6596fb548022ed93f31e30fe'),
  name: 'Punxsutawney ',
  populatiuon: 6200,
  last_sensus: 2008-01-31T00:00:00.000Z,
  famous_for: [
    ''
  ],
  mayor: {
    name: 'Jim Wehrle'
  }
}
{
  _id: ObjectId('6596fb728022ed93f31e30ff'),
  name: 'New York',
  populatiuon: 22200000,
  last_sensus: 2009-07-31T00:00:00.000Z,
  famous_for: [
    'status of liberty',
    'food'
  ],
  mayor: {
    name: 'Michael Bloomberg',
```

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.name" : 1, _id : 0})
< {
    name: 'New York',
    mayor: {
      name: 'Michael Bloomberg'
    }
  }
```

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

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

Практическое задание 3.1.2

1)Сформировать функцию для вывода списка самцов единорогов.

```
myFn = function() {return db.unicorns.find({gender : 'm'})}
[Function: myFn]
myFn
[Function: myFn]
myFn()
{
  _id: ObjectId('6596a3f09dbc95074933cbe7'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId('6596a3f09dbc95074933cbe9'),
  name: 'Unicrom',
  loves: [
    'energon',
    'redbull'
  ]
```

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

3) Вывести результат, используя forEach.

```
> var cursor = myFn().sort({name : 1}).limit(2)
> cursor.forEach(function(obj){print(obj)})
< {
    _id: ObjectId('6596a3f09dbc95074933cbe7'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
< {
    _id: ObjectId('6596a3f09dbc95074933cbed'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
```

Практическое задание 3.2.1

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 3.2.2

Вывести список предпочтений.

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

Практическое задание 3.2.3

Подсчитать количество особей единорогов обоих полов.

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

Практическое задание 3.3.1

Такой команд нету

```
> db.unicorns.save({name: 'Barny', loves: ['grape'],
weight: 340, gender: 'm'})
3 ▸ TypeError: db.unicorns.save is not a function
```

Практическое задание 3.3.2

Для самки единорога `Ayna` *внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.*

Проверить содержимое коллекции `unicorns`.

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

```
  _id: ObjectId('6596a3f09dbc95074933cbeb'),
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],
  weight: 550,
  gender: 'f',
  vampires: 80
}
{
  _id: ObjectId('6596a3f09dbc95074933cbec'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 800,
  gender: 'f',
  vampires: 51
}
```

Практическое задание 3.3.3

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

```
> db.unicorns.update({name : "Raleigh"}, {$set : {"loves" : ["redbull"]}})
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Практическая задание 3.3.4

Всем самцам единорогов увеличить количество убитых вапмиров на 5.

```
> db.unicorns.updateMany({gender : 'm'}, {$inc: {vampires : 5}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 7,
    modifiedCount: 7,
    upsertedCount: 0
  }
learn >
```

Практическое задание 3.3.5

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

```
> db.towns.updateOne({name : "Portland"}, [{$unset : "mayor.party"}])
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Практическое задание 3.3.6

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

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

```
db.unicorns.findOne({name : "Pilot"})
{
  _id: ObjectId('6596a3f09dbc95074933cbf0'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'Chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
```

Практическое задание 3.3.7

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

```
> db.unicorns.updateOne({name : "Aurora"}, { $push : {loves  : {$each : ["sugar" , "lemon"]}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

```
> db.unicorns.findOne({name : "Aurora"})
< {
    _id: ObjectId('6596a3f09dbc95074933cbe8'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar',
      'lemon'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

Практическое задание 3.4.1

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

```
> db.towns.deleteMany({"mayor.party" : {$exists : false}})
{
  acknowledged: true,
  deletedCount: 2
}
```

```
> db.towns.find()
< {
    _id: ObjectId('6596fb728022ed93f31e30ff'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [
      'status of liberty',
      'food'
    ],
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
```

```
> db.towns.deleteMany({})
< {
    acknowledged: true,
    deletedCount: 1
  }
learn>
```

```
> db.towns.drop()
< true
> show collections
< unicorns
```

Практическое задание 4.1.1

Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.

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

Проверьте содержание коллекции

```
> db.places.insertMany([{"_id" : "candy_town", "name" : "CandyTown", "description" : "coming"}, {"_id" : "chine" , "name" : "Chine", "description" : "Xi Jinping"} ])
< {
  acknowledged: true,
  insertedIds: {
    '0': 'candy_town',
    '1': 'chine'
  }
}
```

```
> var candyId = db.places.findOne({_id : "candy_town"})._id
> var chineId = db.places.findOne({_id : "Chine"})._id
▸ TypeError: Cannot read properties of null (reading '_id')
> var chineId = db.places.findOne({_id : "chine"})._id
```

```
> var chineId = db.places.findOne({_id : "chine"})._id
> db.unicorns.updateOne({name : "Horny"}, {$set : {places: {$ref : "places", $id: chineId}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.updateOne({name : "Unicrom"}, {$set : {places: {$ref : "places", $id: candy_town}}})
▸ ReferenceError: candy_town is not defined
> db.unicorns.updateOne({name : "Unicrom"}, {$set : {places: {$ref : "places", $id: candyId}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
learn>
```

```
> db.unicorns.find({name : "Horny"})
< {
    _id: ObjectId('6596a3f09dbc95074933cbe7'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 73,
    places: DBRef('places', 'chine')
  }
```

Практическое задание 4.2.1

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

```
> db.unicorns.ensureIndex({"name" : 1} , {"unique" : true})
< [ 'name_1' ]
```

Практическое задание 4.3.1

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, key: { name: 1 }, name: 'name_1', unique: true }
  ]
```

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

```
> db.unicorns.dropIndex("_id_")
▸ MongoServerError: cannot drop _id index
learn>
```

Практическое задание 4.4.1

Создайте объемную коллекцию `numbers`*, задействовав курсор:*

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

Выберите последних четыре документа.

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

Создайте индекс для ключа `value`.

Получите информацию о всех индексах коллекции `numbres`.

Выполните запрос 2.

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

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

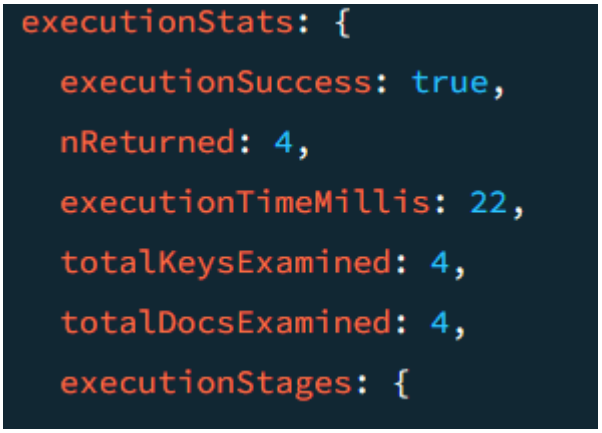
```
> db.createCollection("num")
< { ok: 1 }
> for(i = 0; i < 100000;i++){db.num.insert({value : i})}
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('659925887b95069573d6d0bf')
    }
  }
```

```
> db.num.find().sort({value : -1}).limit(4)
< {
    _id: ObjectId('659925887b95069573d6d0bf'),
    value: 99999
  }
  {
    _id: ObjectId('659925887b95069573d6d0be'),
    value: 99998
  }
  {
    _id: ObjectId('659925887b95069573d6d0bd'),
    value: 99997
  }
  {
    _id: ObjectId('659925877b95069573d6d0bc'),
    value: 99996
  }
```

```
> db.num.find().sort({value : -1}).limit(4).explain("executionStats")
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 133,
```

```
> db.num.ensureIndex({"value" : 1})
< [ 'value_1' ]
> db.num.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { value: 1 }, name: 'value_1' }
  ]
> db.num.find().sort({value : -1}).limit(4).explain("executionStats")
```



Запрос с индексом более эффективный.

Вывод: В этой лабораторной работе были внимательно изучены фундаментальные возможности MongoDB. Практическими заданиями успешно закреплены и усовершенствованы навыки, что способствует более глубокому пониманию принципов работы.