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

Факультет инфокоммуникационных технологий

ЛАБОРАТОРНАЯ РАБОТА №5

РАБОТА С БД В СУБД MongoBD

по дисциплине:
«Проектирование и Реализация Баз Данных»

Выполнила:
студентка II курса ИКТ
группы К3241
Кормановская Д.

Санкт-Петербург
2022

Цель: 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.

CRUD-ОПЕРАЦИИ В СУБД MONGODB. ВСТАВКА ДАННЫХ. ВЫБОРКА ДАННЫХ

ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ

1.1.

Создать базу данных learn, заполнить коллекцию единорогов unicorns, используя два способа, проверить содержимое коллекции.

```
> use learn

> 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'});

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

> db.unicorns.insertOne(document)

> db.unicorns.find()
```

```
< { acknowledged: true,
    insertedIds: { '0': ObjectId("628509974fbbdfb679949312") } }
> 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.insertOne(document)
< { acknowledged: true,
    insertedId: ObjectId("62850a044fbbdfb679949313") }
> db.unicorns.find()
< { _id: ObjectId("628509964fbbdfb679949308"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63 }
  { _id: ObjectId("628509964fbbdfb679949309"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43 }
  { _id: ObjectId("628509964fbbdfb67994930a"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182 }
  { _id: ObjectId("628509964fbbdfb67994930b"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99 }
  { _id: ObjectId("628509964fbbdfb67994930c"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80 }
  { _id: ObjectId("628509974fbbdfb679949312"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f' }
  { _id: ObjectId("628509964fbbdfb67994930d"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40 }
  { _id: ObjectId("628509974fbbdfb67994930e"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39 }
  { _id: ObjectId("628509974fbbdfb67994930f"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2 }
  { _id: ObjectId("628509974fbbdfb679949310"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33 }
  { _id: ObjectId("628509974fbbdfb679949311"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54 }
  { _id: ObjectId("62850a044fbbdfb679949313"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165 }
```

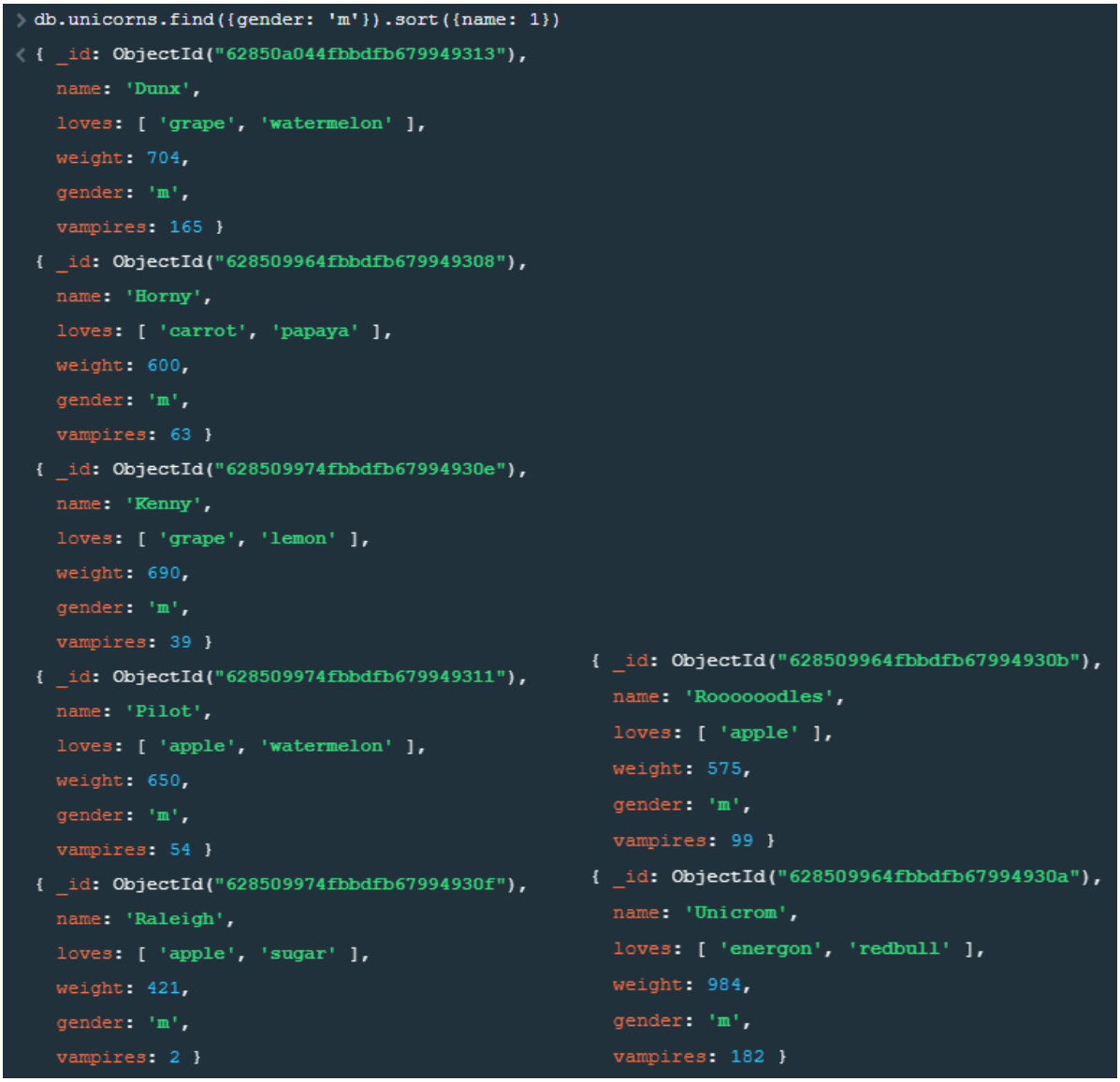
ВЫБОРКА ДАННЫХ ИЗ БД

1.2.

Сформировать запросы:

- *вывод самцов, сортировка по имени*
- *вывод первых трех самок при сортировки по имени*
- *найти всех самку, которая любит carrot(findOne и limit)*

```
> db.unicorns.find({gender: 'm'}).sort({name: 1})
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
```



Выборка самцов (1)

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
< { _id: ObjectId("628509964fbbdfb679949309"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43 }
  { _id: ObjectId("628509964fbbdfb67994930d"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40 }
  { _id: ObjectId("628509974fbbdfb679949310"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33 }
```

Выборка самок (2)

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
< { _id: ObjectId("628509964fbbdfb679949309"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43 }
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
< { _id: ObjectId("628509964fbbdfb679949309"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43 }
```

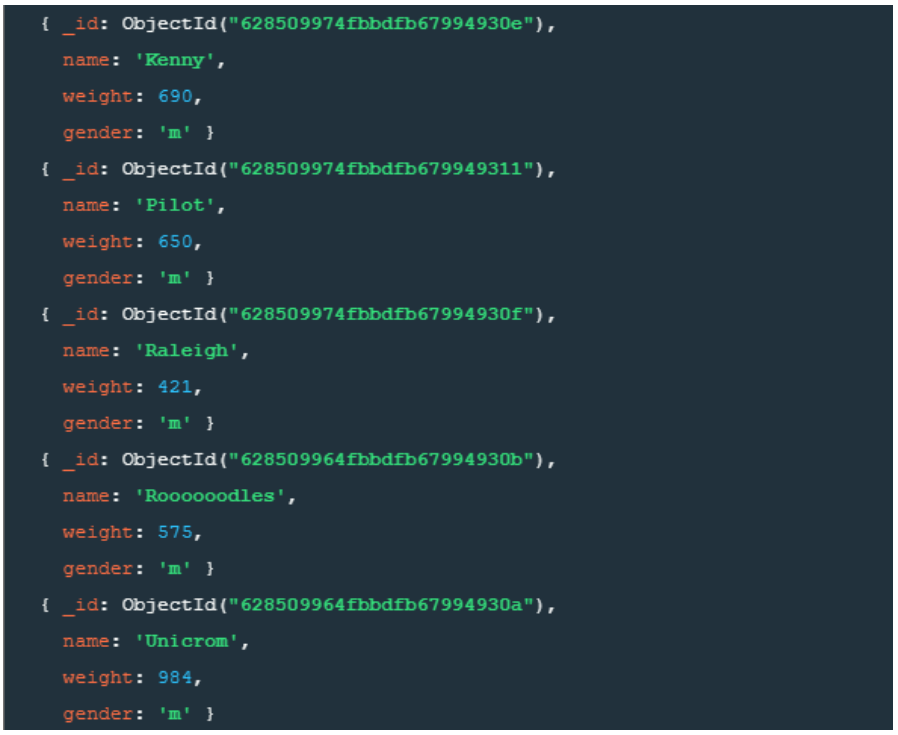
Любительница моркови (3)

1.3.

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

```
> db.unicorns.find({gender: 'm'}, {loves: 0, vampires: 0}).sort({name: 1})
```

```
> db.unicorns.find({gender: 'm'}, {loves: 0, vampires: 0}).sort({name: 1})
< { _id: ObjectId("62850a044fbbdfb679949313"),
    name: 'Dunx',
    weight: 704,
    gender: 'm' }
  { _id: ObjectId("628509964fbbdfb679949308"),
    name: 'Horny',
    weight: 600,
    gender: 'm' }
```



1.4.

Вывести список единорогов в обратном порядке добавления

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

1.5

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

```
> db.unicorns.find({}, {loves: {$slice: 1}, _id: 0})
```

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

1.6.

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

```
> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lt: 700}}, {_id: 0})
```

```
> db.unicorns.find({gender: 'f', weight: {$gte: 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' }
```

1.7.

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

```
> db.unicorns.find({gender: 'm', loves: {$all: ['grape', 'lemon']}, weight: {$gte: 500}}, {_id:
0})
```

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

1.8.

Найти всех единорогов, не имеющих ключ vampires.

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

```
> db.unicorns.find({vampires: {$exists:false}})
< { _id: ObjectId("628509974fbbdfb679949312"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f' }
```

1.9.

Вывести список 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' ] }
```

ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB. ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ

ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ

2.1.

- *Создать коллекцию towns*
- *Сформировать запрос, который возвращает список городов с независимыми мэрами (`party="I"`). Вывести только название города и информацию о мэре.*
- *Сформировать запрос, который возвращает список беспартийных мэров (`party` отсутствует). 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' } }
```

```
> 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' } }
```

ИСПОЛЬЗОВАНИЕ JAVASCRIPT И КУРСОРЫ

2.2.

1. *Сформировать функцию для вывода списка самцов единорогов.*
2. *Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.*
3. *Вывести результат, используя forEach.*

```
> var cursor = db.unicorns.find({gender: 'm'}); null;
> cursor.sort({name: 1}).limit(2); null;
> cursor.forEach(function(obj){ print(obj.name);})
```

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

АГРЕГИРОВАННЫЕ ЗАПРОСЫ

2.3.

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

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

2.4.

Вывести список предпочтений.

```
> db.unicorns.distinct('loves')
```

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

2.5.

Подсчитать 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 }
```

РЕДАКТИРОВАНИЕ ДАННЫХ

2.6.

Выполнить команду, проверить содержимое

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

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

2.7.

Для самки единорога `Ayna` *внести изменения в БД: теперь ее вес* `800`*, она убила* `51` *вампира, проверить содержимое коллекции* `unicorns`*.*

```
> db.unicorns.updateOne({name: 'Ayna'}, {$set:{weight: 800, vampires: 51}})
```

```
> db.unicorns.updateOne({name: 'Ayna'}, {$set:{weight: 800, vampires: 51}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.unicorns.findOne({name: 'Ayna'})
< { _id: ObjectId("628509964fbbdfb67994930d"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51 }
```

2.8.

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

```
db.unicorns.updateOne({name: 'Raleigh'}, {$set:{loves: ['Redbull']}})
```

```
> db.unicorns.updateOne({name: 'Raleigh'}, {$set:{loves: ['Redbull']}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.unicorns.findOne({name: 'Raleigh'})
< { _id: ObjectId("628509974fbbdfb67994930f"),
    name: 'Raleigh',
    loves: [ 'Redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2 }
```

2.9.

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

```
> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
```

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

```
> db.unicorns.find({gender: 'm'})
< { _id: ObjectId("628509964fbbdfb679949308"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68 }
  { _id: ObjectId("628509964fbbdfb67994930a"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187 }
  { _id: ObjectId("628509964fbbdfb67994930b"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104 }
  { _id: ObjectId("628509974fbbdfb67994930e"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 44 }
  { _id: ObjectId("628509974fbbdfb67994930f"),
    name: 'Raleigh',
    loves: [ 'Redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 7 }
  { _id: ObjectId("628509974fbbdfb679949311"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 59 }
  { _id: ObjectId("62850a044fbbdfb679949313"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 170 }
```

2.10.

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

```
> db.towns.updateOne({name: 'Portland'}, {$unset: {'mayor.party': 1}})
```

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

```
> db.towns.find()
< { _id: ObjectId("62948d9092ea9ee693bb9fab"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' } }
  { _id: ObjectId("62948da192ea9ee693bb9fac"),
    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' } }
  { _id: ObjectId("62948db892ea9ee693bb9fad"),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: 2008-01-31T00:00:00.000Z,
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' } }
```

2.11.

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

```
> db.unicorns.updateOne({name: 'Pilot'}, {$push: {loves: 'chocolate'}})
```

```
> db.unicorns.updateOne({name: 'Pilot'}, {$push: {loves: 'chocolate'}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.unicorns.find({name: 'Pilot'})
< { _id: ObjectId("628509974fbbdfb679949311"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59 }
```

2.12.

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

```
> db.unicorns.updateOne({name: 'Aurora'}, {$addToSet: {loves: {$each: ['sugar', 'lemon']}}})
```

```
> db.unicorns.updateOne({name: 'Aurora'}, {$addToSet: {loves: {$each: ['sugar', 'lemon']}}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.unicorns.find({name: 'Aurora'})
< { _id: ObjectId("628509964fbbdfb679949309"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43 }
```

УДАЛЕНИЕ ДАННЫХ ИЗ КОЛЛЕКЦИИ

2.13.

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

```
> db.towns.remove({'mayor.party': {$exists: false}})
```

```
> db.towns.find()
< { _id: ObjectId("6294a2e6d110377d5228bb9b"),
    name: 'Punxsutawney ',
    popujatiuon: 6200,
    last_sensus: 2008-01-31T00:00:00.000Z,
    famous_for: [ 'phil the groundhog' ],
    mayor: { name: 'Jim Wehrle' } }
  { _id: ObjectId("6294a2f4d110377d5228bb9c"),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' } }
  { _id: ObjectId("6294a300d110377d5228bb9d"),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' } }
> db.towns.remove({'mayor.party': {$exists: false}})
< { acknowledged: true, deletedCount: 1 }
> db.towns.find()
< { _id: ObjectId("6294a2f4d110377d5228bb9c"),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' } }
  { _id: ObjectId("6294a300d110377d5228bb9d"),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' } }
> db.towns.remove({})
< { acknowledged: true, deletedCount: 2 }
> show collections
< towns
  unicorns
> db.towns.find()
<
```

ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

ССЫЛКИ В БД

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, используя второй способ автоматического связывания. Проверьте содержание коллекции единорогов.

```
> db.kingdoms.insertOne({_id:'rn', name: 'Rainbowland', description: 'Where all rainbow begins'})
< { acknowledged: true, insertedId: 'rn' }
> db.kingdoms.insertOne({_id:'sg', name: 'Sugara', description: 'The sweetest place in the Universe'})
< { acknowledged: true, insertedId: 'sg' }
> db.unicorns.updateOne({name: 'Roooooodles'}, {$set: {kingdom: {$ref: 'kingdoms', $id: 'rn'}}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.unicorns.updateOne({name: 'Pilot'}, {$set: {kingdom: {$ref: 'kingdoms', $id: 'sg'}}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.unicorns.find({kingdom: {$exists: true}})
< { _id: ObjectId("628509964fbbdfb67994930b"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104,
    kingdom: DBRef("kingdoms", 'rn') }
  { _id: ObjectId("628509974fbbdfb679949311"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59,
    kingdom: DBRef("kingdoms", 'sg') }
```

НАСТРОЙКА ИНДЕКСОВ

3.2.

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

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

УПРАВЛЕНИЕ ИНДЕКСАМИ

3.3.

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

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

ПЛАН ЗАПРОСА

3.4.

Создайте объемную коллекцию `numbers`*.Выберите последних четыре документа. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра* `executionTimeMillis`*)*

```
> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
< 'DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.'
< { acknowledged: true,
    insertedIds: { '0': ObjectId("6294aa3dd110377d522a423d") } }
> db.numbers.find().limit(15)
< { _id: ObjectId("6294a972d110377d5228bb9e"), value: 0 }
  { _id: ObjectId("6294a972d110377d5228bb9f"), value: 1 }
  { _id: ObjectId("6294a972d110377d5228bba0"), value: 2 }
  { _id: ObjectId("6294a972d110377d5228bba1"), value: 3 }
  { _id: ObjectId("6294a972d110377d5228bba2"), value: 4 }
  { _id: ObjectId("6294a972d110377d5228bba3"), value: 5 }
  { _id: ObjectId("6294a972d110377d5228bba4"), value: 6 }
  { _id: ObjectId("6294a972d110377d5228bba5"), value: 7 }
  { _id: ObjectId("6294a972d110377d5228bba6"), value: 8 }
  { _id: ObjectId("6294a972d110377d5228bba7"), value: 9 }
  { _id: ObjectId("6294a972d110377d5228bba8"), value: 10 }
  { _id: ObjectId("6294a972d110377d5228bba9"), value: 11 }
  { _id: ObjectId("6294a972d110377d5228bbaa"), value: 12 }
  { _id: ObjectId("6294a972d110377d5228bbab"), value: 13 }
  { _id: ObjectId("6294a972d110377d5228bbac"), value: 14 }
> db.numbers.explain('executionStats').find().sort({value: -1}).limit(4)
```

```
executionTimeMillis: 78,
```

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

```
executionTimeMillis: 18,
```

Второй запрос в разы эффективнее

Выводы: выполнена работа с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.