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

Отчет

по лабораторной работе №6 «Процедуры, функции, триггеры в PostgreSQL»

по дисциплине **«Работа с БД в СУБД MongoDB»**

Автор: Бунос М.В.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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.

Оборудование: компьютерный класс.

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

Создайте базу данных learn.

```
rs01 [primary]> use learn
switched to db learn
```

Заполните коллекцию единорогов unicorns:

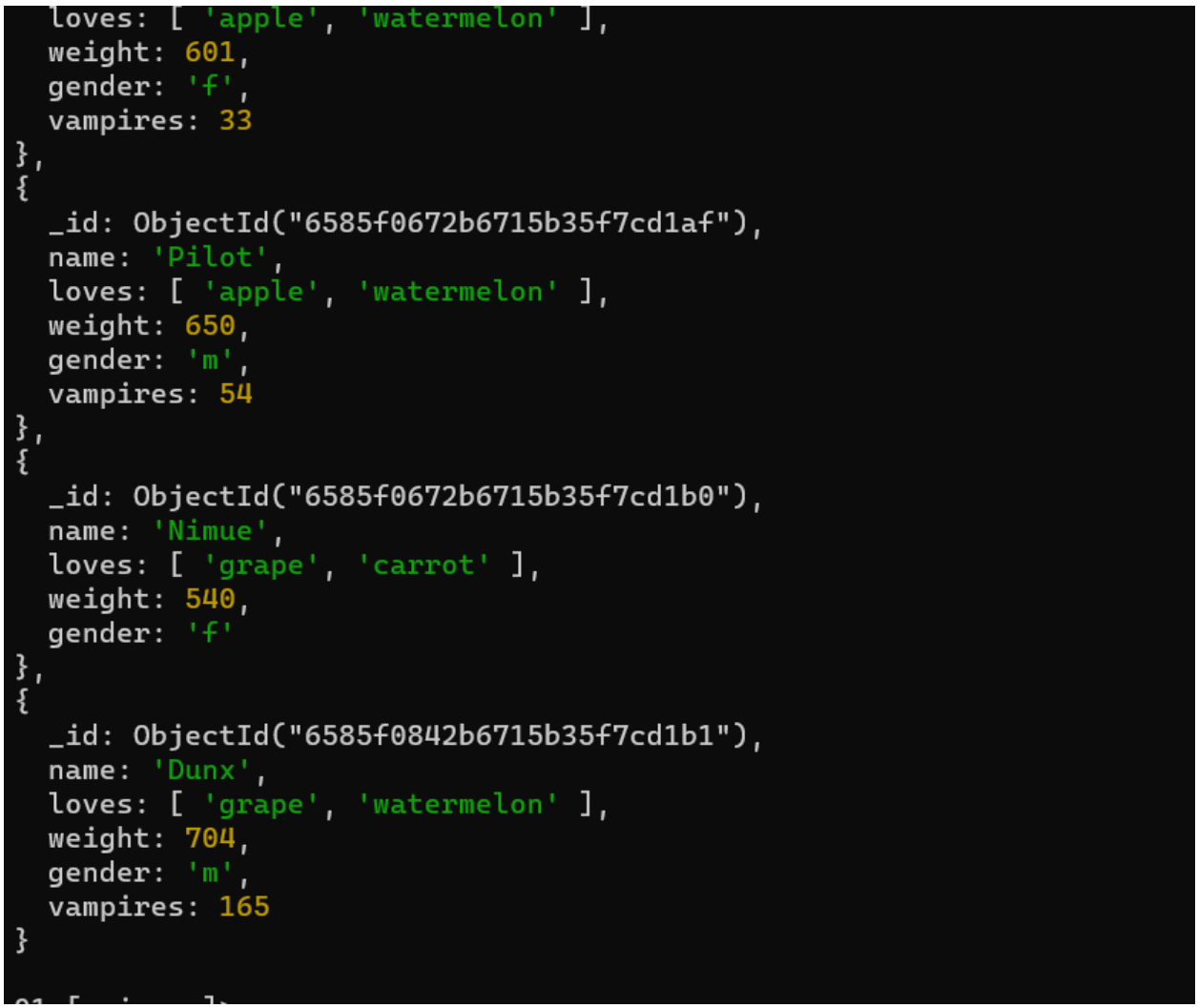
```
db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: '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'});
```

Используя второй способ, вставьте в коллекцию единорогов документ:

```
{name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
```

```
rs01 [primary]>
rs01 [primary]> db.unicorns.insert({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6585f0842b6715b35f7cd1b1") }
}
```

Проверьте содержимое коллекции с помощью метода find.



2.2 ВЫБОРКА ДАННЫХ ИЗ БД

Практическое задание 2.2.1:

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

```
rs01 [primary]> db.unicorns.find({gender: 'm'}).sort({name: 1}).limit
[
  {
    _id: ObjectId("6585f0842b6715b35f7cd1b1"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("6585f0662b6715b35f7cd1a6"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("6585f0672b6715b35f7cd1ac"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

```
rs01 [primary]> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit
[
  {
    _id: ObjectId("6585f0662b6715b35f7cd1a7"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("6585f0672b6715b35f7cd1ab"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId("6585f0672b6715b35f7cd1ae"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
```

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

```
]
rs01 [primary]> db.unicorns.find({gender: 'f', loves: 'carrot'});
[
  {
    _id: ObjectId("6585f0662b6715b35f7cd1a7"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("6585f0672b6715b35f7cd1aa"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("6585f0672b6715b35f7cd1b0"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

```
rs01 [primary]> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(
[
  {
    _id: ObjectId("6585f0662b6715b35f7cd1a7"),
    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 предпотениях и поле.

```
rs01 [primary]> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
[
  {
    _id: ObjectId("6585f0662b6715b35f7cd1a6"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId("6585f0662b6715b35f7cd1a8"),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId("6585f0662b6715b35f7cd1a9"),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

```
rs01 [primary]> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId("6585f0842b6715b35f7cd1b1"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("6585f0672b6715b35f7cd1b0"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
```

Практическое задание 2.1.4:

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

```
rs01 [primary]> db.unicorns.find({}, {_id: 0, loves: {$slice: [0, 1]}})
[
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Aurora',
    loves: [ 'carrot' ],
    weight: 450,
    gender: 'f',
    vampires: 43
```

2.3 ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Практическое задание 2.3.1:

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

```
rs01 [primary]> db.unicorns.find({weight: {$lt: 700, $gt: 500}, gender: 'f'}, {_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'
  }
]
```

Практическое задание 2.3.2:

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

```
rs01 [primary]> db.unicorns.find({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`.

```
rs01 [primary]> db.unicorns.find({vampires: {$exists: false}});
[
  {
    _id: ObjectId("6585f0672b6715b35f7cd1b0"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 2.3.4:

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

```
rs01 [primary]> db.unicorns.find({gender: 'm'}, {name: 1, _id: 0, loves: {$s
lice: [0, 1]}});
[
  { name: 'Horny', loves: [ 'carrot' ] },
  { name: 'Unicrom', loves: [ 'energon' ] },
  { name: 'Roooooodles', loves: [ 'apple' ] },
  { name: 'Kenny', loves: [ 'grape' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Pilot', loves: [ 'apple' ] },
  { name: 'Dunx', loves: [ 'grape' ] }
]
```

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

3.1 ЗАПРОС К ВЛОЖЕННЫМ ОБЪЕКТАМ

Практическое задание 3.1.1:

1. *Создайте коллекцию towns, включающую следующие документы:*

```
{name: "Punxsutawney ",
populatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: [""],
mayor: {
   name: "Jim Wehrle"
   }}

{name: "New York",
populatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
   name: "Michael Bloomberg",
   party: "I"}}

{name: "Portland",
```

```
        populatiuon: 528000,
        last_sensus: ISODate("2009-07-20"),
        famous_for: ["beer", "food"],
        mayor: {
            name: "Sam Adams",
            party: "D"}}
```

2. Сформировать запрос, который возвращает список городов с независимыми мэрами (`party="I"`*). Вывести только название города и информацию о мэре.*

```
rs01 [primary]> db.towns.find({'mayor.party': 'I'}, {'name': 1, 'mayor.name'
: 1})
[
  {
    _id: ObjectId("6585f2e52b6715b35f7cd1b6"),
    name: 'New York',
    mayor: { name: 'Michael Bloomberg' }
  }
]
rs01 [primary]>
```

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

```
rs01 [primary]> db.towns.find({'mayor.party': {$exists: false}}, {'name': 1,
 'mayor.name': 1})
[
  {
    _id: ObjectId("6585f2d82b6715b35f7cd1b5"),
    name: 'Punxsutawney ',
    mayor: { name: 'Jim Wehrle' }
  }
]
rs01 [primary]>
```

Практическое задание 3.1.2:

3. Сформировать функцию для вывода списка самцов единорогов.

```
rs01 [primary]> function getMaleUnicorns() {
... return db.unicorns.find({gender: 'm'})
... }
[Function: getMaleUnicorns]
rs01 [primary]> getMaleUnicorns();
[
  {
    _id: ObjectId("6585f0662b6715b35f7cd1a6"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("6585f0662b6715b35f7cd1a8"),
    name: 'Unicrom'
```

4. *Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.*

```
rs01 [primary]> var cursor = getMaleUnicorns().sort({name: 1}).limit(2);
```

```
rs01 [primary]> cursor.forEach(function(obj){print(obj)});
```

5. *Вывести результат, используя forEach.*

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
rs01 [primary]> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 600
}}).count();
2
```

Практическое задание 3.2.2:

Вывести список предпочтений.

```
rs01 [primary]> db.unicorns.distinct('loves');
[
  'apple',      'carrot',
  'chocolate',  'energon',
  'grape',      'lemon',
  'papaya',     'redbull',
  'strawberry', 'sugar',
  'watermelon'
]
rs01 [primary]>
```

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

```
rs01 [primary]> db.unicorns.aggregate({$group: {_id: '$gender', count: {$sum
: 1}}});
[ { _id: 'm', count: 6 }, { _id: 'f', count: 5 } ]
```

Практическое задание 3.3.1:

1. *Выполнить команду:*

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

```
MongoshInvalidInputError: Collection.save() is deprecated. Use insertOne, in
sertMany, updateOne, or updateMany.
rs01 [primary]>

rs01 [primary]> db.unicorns.insert({name: 'Barny', loves: ['grape'], weight:
 340, gender: 'm'})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6585f63e2b6715b35f7cd1ce") }
}
```

2. *Проверить содержимое коллекции* `unicorns`.

```
  {
    _id: ObjectId("6585f63e2b6715b35f7cd1ce"),
    name: 'Barny',
    loves: [ 'grape' ],
    weight: 340,
    gender: 'm'
  }
]
rs01 [primary]>
```

Практическое задание 3.3.2:

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

```
rs01 [primary]> db.unicorns.updateOne({name: 'Ayna'}, [{$set: {weight: 500,
vampires: 51}}])
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
rs01 [primary]> db.unicorns.findOne({name: 'Ayna'})
{
  _id: ObjectId("6585f4f72b6715b35f7cd1c8"),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 500,
  gender: 'f',
  vampires: 51
}
```

2. *Проверить содержимое коллекции* `unicorns`

Практическое задание 3.3.3:

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

```
rs01 [primary]> db.unicorns.updateOne({name: 'Raleigh'}, [{$set: {'loves'
'redbull']}}])
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
rs01 [primary]> db.unicorns.findOne({name: 'Raleigh'})
{
  _id: ObjectId("6585f4f72b6715b35f7cd1ca"),
  name: 'Raleigh',
  loves: [ 'redbull' ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
rs01 [primary]>
```

Практическое задание 3.3.4:

1. *Всем самцам единорогов увеличить количество убитых вапмиров на 5.*

```
 s01 [primary]> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
rs01 [primary]> db.unicorns.find()
[
  {
    _id: ObjectId("6585f4f72b6715b35f7cd1c3"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId("6585f4f72b6715b35f7cd1c4"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("6585f4f72b6715b35f7cd1c5")
```

2. *Проверить содержимое коллекции* `unicorns`.

Практическое задание 3.3.5:

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

```
rs01 [primary]> db.towns.updateOne({name: 'Portland'}, [{$unset: 'mayor.part
y'}]);
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
rs01 [primary]> do.towns.find({name: 'Portland'})
Uncaught:
SyntaxError: Unexpected token (1:2)

> 1 | do.towns.find({name: 'Portland'})
    |   ^
  2 |

rs01 [primary]> db.towns.find({name: 'Portland'})
[
  {
    _id: ObjectId("6585f2ee2b6715b35f7cd1b7"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
rs01 [primary]>
```

2. *Проверить содержимое коллекции* `towns`.

Практическое задание 3.3.6:

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

```
rs01 [primary]> db.unicorns.updateOne({name: 'Pilot'}, {$push: {loves: 'choc
olate'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
rs01 [primary]> db.unicorns.find({name: 'Pilot'})
[
  {
    _id: ObjectId("6585f4f72b6715b35f7cd1cc"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

2. *Проверить содержимое коллекции* `unicorns`.

Практическое задание 3.3.7:

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

```
rs01 [primary]> db.unicorns.updateOne({name: 'Aurora'}, {$push: {loves: {$ea
ch: ['sugar', 'lemon']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
rs01 [primary]> db.unicorns.find({name: 'Aurora'})
[
  {
    _id: ObjectId("6585f4f72b6715b35f7cd1c4"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
rs01 [primary]>
```

2. *Проверить содержимое коллекции* `unicorns`.

Практическое задание 3.4.1:

4. *Создайте коллекцию towns, включающую следующие документы:*

```
{name: "Punxsutawney ",
popujatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: ["phil the groundhog"],
mayor: {
   name: "Jim Wehrle"
   }}

{name: "New York",
popujatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
   name: "Michael Bloomberg",
   party: "I"}}

{name: "Portland",
popujatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
   name: "Sam Adams",
   party: "D"}}
```

5. *Удалите документы с беспартийными мэрами.*

```
rs01 [primary]> db.towns.deleteMany({'mayor.party': {$exists: false}
{ acknowledged: true, deletedCount: 1 }
rs01 [primary]> db.towns.find()
[
  {
    _id: ObjectId("6585fc2e2b6715b35f7cd1d5"),
    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("6585fc322b6715b35f7cd1d6"),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

6. *Проверьте содержание коллекции.*
7. *Очистите коллекцию.*
8. *Просмотрите список доступных коллекций.*

```
rs01 [primary]> db.towns.drop();
true
rs01 [primary]> db.towns.find();

rs01 [primary]> show collections
unicorns
users
```

4 ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

4.1 ССЫЛКИ В БД

Практическое задание 4.1.1:

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

```
rs01 [primary]> db.locations.insertMany([{"_id": "LUN", "name": "The Luna",
"description": "lol"}, {"_id": "SUN", "name": "The Sun", "description": "kek
"}, {"_id": "EARTH", "name": "The Earth", "description": "lmao"}
... ])
{
  acknowledged: true,
```

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

```
rs01 [primary]> db.unicorns.updateMany({loves: {$in: ['grape']}}, {$set: {lo
cation: {$ref: 'locations', $id: 'LUN'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
rs01 [primary]>
```

9. *Проверьте содержание коллекции единорогов.*

```
rs01 [primary]> db.unicorns.find({loves: {$in: ['grape']}})
[
  {
    _id: ObjectId("6585f4f72b6715b35f7cd1c4"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    location: DBRef("locations", "LUN")
  },
  {
    _id: ObjectId("6585f4f72b6715b35f7cd1c9"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 44,
    location: DBRef("locations", "LUN")
  },
  {
    _id: ObjectId("6585f4f72b6715b35f7cd1cd"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
```

4.3 УПРАВЛЕНИЕ ИНДЕКСАМИ

Практическое задание 4.3.1:

11. *Получите информацию о всех индексах коллекции* `unicorns` .

```
rs01 [primary]> db.unicorns.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
rs01 [primary]>
```

12. *Удалите все индексы, кроме индекса для идентификатора.*

```
rs01 [primary]> db.unicorns.dropIndex('name_1')
{
  nIndexesWas: 2,
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp(1, 1703280821),
    signature: {
      hash: Binary(Buffer.from("c0864c780c1cc095cc3d7d2c49ebc855419fec2e", "
hex"), 0),
      keyId: Long("7315507586415984646")
    }
  },
  operationTime: Timestamp(1, 1703280821)
}
```

13. Попытайтесь удалить индекс для идентификатора.

```
rs01 [primary]> db.unicorns.dropIndex('_id_')
MongoError: cannot drop _id index
```

4.4 ПЛАН ЗАПРОСА

Практическое задание 4.4.1:

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

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

2. *Выберите последних четыре документа*

```
[learn> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
;
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('658348f9db12a495f317eabc') }
}
learn> ;

[learn> db.numbers.find().count();
100000
[learn> db.numbers.find({value: {$in: [9996, 9997, 9998, 9999]}})
[
  { _id: ObjectId('658348d2db12a495f3168b29'), value: 9996 },
  { _id: ObjectId('658348d2db12a495f3168b2a'), value: 9997 },
  { _id: ObjectId('658348d2db12a495f3168b2b'), value: 9998 },
  { _id: ObjectId('658348d2db12a495f3168b2c'), value: 9999 }
]
```

.

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

4. *Создайте индекс для ключа* `value`*.*

```
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 63,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
```

5. Получите информацию о всех индексах коллекции `numbres`.

6. Выполните запрос 2.

```
learn> db.numbers.ensureIndex({value: 1})
[ 'value_1' ]
learn> db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
learn> db.numbers.find({value: {$in: [9996, 9997, 9998, 9999]}})
[
  { _id: ObjectId('658348d2db12a495f3168b29'), value: 9996 },
  { _id: ObjectId('658348d2db12a495f3168b2a'), value: 9997 },
  { _id: ObjectId('658348d2db12a495f3168b2b'), value: 9998 },
  { _id: ObjectId('658348d2db12a495f3168b2c'), value: 9999 }
]
```

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

```
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 1,
    totalKeysExamined: 5,
    totalDocsExamined: 4,
```

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

Ускорение с индексом очень сильно заметно.

Вывод

В ходе лабораторной работы была изучена работа с NoSQL БД MongoDB.