

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО»**

Факультет инфокоммуникационных технологий

Дисциплина:

«Проектирование и реализация баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5.2

«Работа в с БД в СУБД MongoDB»

Выполнил:

студент группы К32392

Байков Иван

(подпись)

Проверил(а):

Говорова Марина Михайловна

(отметка о выполнении)

(подпись)

Санкт-Петербург
2023 г.

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

Практическое задание: привести результаты выполнения практических заданий (номер задания, формулировка, команда, лог (скриншот) результата, вывод (при необходимости)).

Выполнение работы: для выполнения работы было предложено воспользоваться текущей версией ПО (6.0.6), однако из-за возникших при установке проблем в работе была использована версия 5.0.17

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

ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ

Практическое задание 8.1.1:

1. Создайте базу данных learn.

```
test> use learn  
switched to db learn
```

2. Заполните коллекцию единорогов unicorns:

```

learn> db.unicorns.insert({name: 'Horny', loves: ['carrot', 'papaya'], weight: 600, gender: 'm',
icorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender:
... vampires: 63});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a4fb0e5aba34a3f5e9ad") }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f',
... vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a4fb0e5aba34a3f5e9ae") }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender:
... 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a4fb0e5aba34a3f5e9af") }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm',
... vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a4fb0e5aba34a3f5e9b0") }
}
learn> db.unicorns.insert({name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550,
... gender: 'f', vampires: 80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a4fc0e5aba34a3f5e9b1") }
}
learn> db.unicorns.insert({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f',
... vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a4fc0e5aba34a3f5e9b2") }
}
learn> db.unicorns.insert({name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm',
... vampires: 39});

```

```

{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a4fc0e5aba34a3f5e9b3") }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm',
... vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a4fc0e5aba34a3f5e9b4") }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f',
... vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a4fc0e5aba34a3f5e9b5") }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender:
... 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6477a55c0e5aba34a3f5e9b6") }
}

```

3. Используя второй способ, вставьте в коллекцию единорогов документ:

```

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

```

4. Проверьте содержимое коллекции с помощью метода `find`.

```
{
  _id: ObjectId("64779cbaf30565ea749c40ea"),
  name: 'Leia',
  loves: [ 'apple', 'watermelon' ],
  weight: 601,
  gender: 'f',
  vampires: 33
},
{
  _id: ObjectId("64779cc3f30565ea749c40eb"),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon' ],
  weight: 650,
  gender: 'm',
  vampires: 54
},
{
  _id: ObjectId("64779cccf30565ea749c40ec"),
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f'
},
{
  _id: ObjectId("64779debf30565ea749c40ed"),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
]
```

ВЫБОРКА ДАННЫХ ИЗ БД

Практическое задание 8.1.2:

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

```

learn> db.unicorns.find({gender: 'm'}).limit(3).sort({name: 1})
[
  {
    _id: ObjectId("6477a8f80e5aba34a3f5e9b8"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9ad"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("6477a4fc0e5aba34a3f5e9b3"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
learn> db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
[
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9ae"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("6477a4fc0e5aba34a3f5e9b2"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId("6477a4fc0e5aba34a3f5e9b5"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
  }
]

```

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

```
[learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
[
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9ae"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
[learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId("6477a4fb0e5aba34a3f5e9ae"),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Практическое задание 8.1.3:

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

```

[learn> db.unicorns.find({gender: 'm'}, {gender: 0, loves: 0})
[
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9ad"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9af"),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9b0"),
    name: 'Rooooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId("6477a4fc0e5aba34a3f5e9b3"),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId("6477a4fc0e5aba34a3f5e9b4"),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId("6477a55c0e5aba34a3f5e9b6"),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId("6477a8f80e5aba34a3f5e9b8"),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  }
]

```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления

```
learn> db.unicorns.find({}).sort({$natural: -1})
[
  {
    _id: ObjectId("6477a8f80e5aba34a3f5e9b8"),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId("6477a5fc0e5aba34a3f5e9b7"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("6477a55c0e5aba34a3f5e9b6"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId("6477a4fc0e5aba34a3f5e9b5"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId("6477a4fc0e5aba34a3f5e9b4"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId("6477a4fc0e5aba34a3f5e9b3"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
  }
]
```



```

},
{
  _id: ObjectId("6477a4fc0e5aba34a3f5e9b2"),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 733,
  gender: 'f',
  vampires: 40
},
{
  _id: ObjectId("6477a4fc0e5aba34a3f5e9b1"),
  name: 'Solnara',
  loves: [ 'apple', 'carrot', 'chocolate' ],
  weight: 550,
  gender: 'f',
  vampires: 80
},
{
  _id: ObjectId("6477a4fb0e5aba34a3f5e9b0"),
  name: 'Rooooooodles',
  loves: [ 'apple' ],
  weight: 575,
  gender: 'm',
  vampires: 99
},
{
  _id: ObjectId("6477a4fb0e5aba34a3f5e9af"),
  name: 'Unicrom',
  loves: [ 'energon', 'redbull' ],
  weight: 984,
  gender: 'm',
  vampires: 182
},
{
  _id: ObjectId("6477a4fb0e5aba34a3f5e9ae"),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
  _id: ObjectId("6477a4fb0e5aba34a3f5e9ad"),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
]

```

Практическое задание 8.1.5:

Вывести список единорогов с названием первого любимого
предпочтения, исключив идентификатор

```
[learn> 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: 'Rooooooodles',
    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',
```

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Практическое задание 8.1.6:

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

```
learn> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 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'
  }
]
```

Практическое задание 8.1.7:

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

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

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ vampires.

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId("6477a5fc0e5aba34a3f5e9b7"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 8.1.9:

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: {$slice: 1}, name: true, _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

Практическое задание 8.2.1:

1. Создайте коллекцию towns, включающую следующие документы:

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

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

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

```

learn> db.towns.insert([
...   {
...     name: "Punxsutawney",
...     population: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: [],
...     mayor: {
...       name: "Jim Wehrle"
...     }
...   },
...   {
...     name: "New York",
...     population: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["status of liberty", "food"],
...     mayor: {
...       name: "Michael Bloomberg",
...       party: "I"
...     }
...   },
...   {
...     name: "Portland",
...     population: 528000,
...     last_sensus: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: {
...       name: "Sam Adams",
...       party: "D"
...     }
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6477acd60e5aba34a3f5e9b9"),
    '1': ObjectId("6477acd60e5aba34a3f5e9ba"),
    '2': ObjectId("6477acd60e5aba34a3f5e9bb")
  }
}

```

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

```
learn> 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.

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

КУРСОРЫ

Практическое задание 8.2.2:

1. Сформировать функцию для вывода списка самцов единорогов.

```
learn> male = function() {return this.gender == 'm'}
[Function: male]
```

2. Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.
3. Вывести результат, используя forEach.

```
learn> var cursor = db.unicorns.find({ $where: function() { return this.gender === 'm'; } }).limit(2).sort({ name: 1 });
```

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



```
4. 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:
'Rooooooodles', 44), 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'});
db.unicorns.insert ({ name: 'Dunx', loves: ['grape', 'watermelon'], weight:
704, gender: 'm', vampires: 165})
```

АГРЕГИРОВАННЫЕ ЗАПРОСЫ

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

Практическое задание 8.2.4:

Вывести список предпочтений.


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

Практическое задание 8.2.5:

Подсчитать количество особей единорогов обоих полов.

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

РЕДАКТИРОВАНИЕ ДАННЫХ

1. Выполнить команду:

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

```
[learn> db.unicorns.insertOne({ name: 'Barney', loves: ['grape'], weight: 340, gender: 'm' })
{
  acknowledged: true,
  insertedId: ObjectId("6477b0b90e5aba34a3f5e9bc")
}
```

1)

2. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.find({})
[
  {
    _id: ObjectId("64779c85f30565ea749c40e2"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("64779c4ef30565ea749c40e3"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("64779c5af30565ea749c40e4"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("64779c6bf30565ea749c40e5"),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("64779c7bf30565ea749c40e6"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("64779c8ef30565ea749c40e7"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId("64779c9af30565ea749c40e8"),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId("64779caef30565ea749c40e9"),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId("64779cbaf30565ea749c40ea"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId("64779cc3f30565ea749c40eb"),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
]
```

```
{
  _id: ObjectId("64779cccf30565ea749c40ec"),
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f'
},
{
  _id: ObjectId("64779dbf30565ea749c40ed"),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
},
{
  _id: ObjectId("6477bba0f30565ea749c40f1"),
  name: 'Barney',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm'
}
]
```

Практическое задание 8.2.7:

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

```
learn> db.unicorns.updateOne({ name: 'Ayna' }, {
...   $set: {
...     name: 'Ayna',
...     loves: ['strawberry', 'lemon'],
...     weight: 800,
...     gender: 'f'
...   }
... });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

2. Проверить содержимое коллекции unicorns.

```
{
  _id: ObjectId("6477a4fc0e5aba34a3f5e9b2"),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 800,
  gender: 'f',
  vampires: 40
},
```

Практическое задание 8.2.8:

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

```
learn> db.unicorns.updateOne({ name: 'Raleigh' }, { $set: { loves: ['redbull']} })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
{
  _id: ObjectId("6477a4fc0e5aba34a3f5e9b4"),
  name: 'Raleigh',
  loves: [ 'redbull' ],
  weight: 421,
  gender: 'm',
  vampires: 2
},
```

Практическое задание 8.2.9:

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.

```
[learn> db.unicorns.update({gender: 'm'}, {$inc: {vampires: 5}}, {multi: true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
```

2. Проверить содержимое коллекции unicorns.

```

[learn> db.unicorns.find()
[
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9ad"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9ae"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9af"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId("6477a4fb0e5aba34a3f5e9b0"),
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104
  },
  {
    _id: ObjectId("6477a4fc0e5aba34a3f5e9b1"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,

```

Практическое задание 8.2.10:

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

```

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

```

2. Проверить содержимое коллекции towns.

```
learn> db.towns.find({})
[
  {
    _id: ObjectId("6477a78ff30565ea749c40ee"),
    name: 'Punxsutawney',
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId("6477a84ff30565ea749c40ef"),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: ISODate("2009-07-31T00:00:00.000Z"),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId("6477a88ff30565ea749c40f0"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

Практическое задание 8.2.11:

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

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

2. Проверить содержимое коллекции unicorns.

```
{
  _id: ObjectId("6477a55c0e5aba34a3f5e9b6"),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon', 'chocolate' ],
  weight: 650,
  gender: 'm',
  vampires: 59
},
```

Практическое задание 8.2.12:

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

```
learn> db.unicorns.update({name: 'Aurora'}, {$addToSet: {loves: {$each: ['sugar', 'lemon']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
{
  _id: ObjectId("6477a4fb0e5aba34a3f5e9ae"),
  name: 'Aurora',
  loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
```

УДАЛЕНИЕ ДАННЫХ ИЗ КОЛЛЕКЦИИ

Практическое задание 8.2.13:

1. Создайте коллекцию towns, включающую следующие документы:


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



```
{name: "New York", population: 22200000, last_sensus:
ISODate("2009-07- 31"), famous_for: ["status of liberty", "food"],
mayor:
{name: "Michael Bloomberg", party: "I"}} {name: "Portland",
population: 528000, last_sensus: ISODate("2009-07- 20"),
famous_for: ["beer", "food"], mayor: {name: "Sam Adams", party:
"D"}}
```

```
learn> db.towns.insert([
...   {
...     name: "Punxsutawney",
...     population: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: ["phil the groundhog"],
...     mayor: { name: "Jim Wehrle" }
...   },
...   {
...     name: "New York",
...     population: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["status of liberty", "food"],
...     mayor: { name: "Michael Bloomberg", party: "I" }
...   },
...   {
...     name: "Portland",
...     population: 528000,
...     last_sensus: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: { name: "Sam Adams", party: "D" }
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6477b5950e5aba34a3f5e9c0"),
    '1': ObjectId("6477b5950e5aba34a3f5e9c1"),
    '2': ObjectId("6477b5950e5aba34a3f5e9c2")
  }
}
```

2. Удалите документы с беспартийными мэрами.

3. Проверьте содержание коллекции

```
[learn> db.towns.deleteMany({'mayor.party': {'$exists': 0}})
{ acknowledged: true, deletedCount: 1 }
[learn> db.towns.find({})
[
  {
    _id: ObjectId("6477b5950e5aba34a3f5e9c1"),
    name: 'New York',
    population: 22200000,
    last_sensus: ISODate("2009-07-31T00:00:00.000Z"),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId("6477b5950e5aba34a3f5e9c2"),
    name: 'Portland',
    population: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

4. Очистите коллекцию.

```
[learn> db.towns.drop();
true
```

5. Просмотрите список доступных коллекций.

```
[learn> show collections
unicorns
```

ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

ССЫЛКИ В БД

Практическое задание 8.3.1:

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

```
learn> db.areas.insert({'_id': 'rus', 'name': 'Russia', 'description': "There are lots of unicorns" });
{ acknowledged: true, insertedIds: { '0': 'rus' } }
```

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

```
[learn> db.unicorns.update({name: 'Raleigh'}, {$set: {area: {$ref: 'areas', $id: 'rus'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

3. Проверьте содержание коллекции единорогов.

```
{
  _id: ObjectId("6477a4fc0e5aba34a3f5e9b4"),
  name: 'Raleigh',
  loves: [ 'redbull' ],
  weight: 421,
  gender: 'm',
  vampires: 7,
  area: DBRef("areas", 'rus')
},
{
```

НАСТРОЙКА ИНДЕКСОВ

Практическое задание 8.3.2:

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

```
[learn> db.unicorns.createIndex({'name': 1}, {'unique': true})
name_1
```

Вывод: для коллекции можно задать индекс для ключа name с флагом unique, так как до этого в коллекции имена единорогов не повторялись.

УПРАВЛЕНИЕ ИНДЕКСАМИ

Практическое задание 8.3.3:

1. Получите информацию о всех индексах коллекции unicorns .

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

2. Удалите все индексы, кроме индекса для идентификатора.

```
[learn> db.unicorns.dropIndex('name_1')  
{ nIndexWas: 2, ok: 1 }
```

Индекс удаляется по своему имени, которое можно посмотреть после выполнения операции `getIndexes()`, а не по названию поля, которому соответствует индекс. Поэтому в качестве аргумента для `dropIndex()` используется строка `'name_1'`, соответствующая названию индекса для ключа `name`

3. Попробуйте удалить индекс для идентификатора.

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

Уникальный индекс для идентификатора `_id` удалить нельзя (см. соответствующее сообщение об ошибке). Об этом же говорит и официальная документация MongoDB.

Практическое задание 8.3.4:

1. Создайте объемную коллекцию numbers, задействовав курсор:

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

```
MongoServerError: cannot drop _id index  
[learn> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}  
  
{  
  acknowledged: true,  
  insertedIds: { '0': ObjectId("6477b9e20e5aba34a3f77062") }  
}
```

3. Выберите последних четыре документа.

```
[learn> db.numbers.find({}).skip(99996)
[
  { _id: ObjectId("6477b9e20e5aba34a3f7705f"), value: 99996 },
  { _id: ObjectId("6477b9e20e5aba34a3f77060"), value: 99997 },
  { _id: ObjectId("6477b9e20e5aba34a3f77061"), value: 99998 },
  { _id: ObjectId("6477b9e20e5aba34a3f77062"), value: 99999 }
]
```

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

```
[learn> db.numbers.explain('executionStats').find({}).skip(99996)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'SKIP',
      skipAmount: 0,
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 43,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      stage: 'SKIP',
      nReturned: 4,
      executionTimeMillisEstimate: 1,
      works: 100002,
      advanced: 4,
      needTime: 99997,
      needYield: 0,
      saveState: 100,
      restoreState: 100,
      isEOF: 1,
      skipAmount: 0,

```

```

    executionTimeMillis: 43,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      stage: 'SKIP',
      nReturned: 4,
      executionTimeMillisEstimate: 1,
      works: 100002,
      advanced: 4,
      needTime: 99997,
      needYield: 0,
      saveState: 100,
      restoreState: 100,
      isEOF: 1,
      skipAmount: 0,
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 100000,
        executionTimeMillisEstimate: 1,
        works: 100002,
        advanced: 100000,
        needTime: 1,
        needYield: 0,
        saveState: 100,
        restoreState: 100,
        isEOF: 1,
        direction: 'forward',
        docsExamined: 100000
      }
    }
  },
  command: { find: 'numbers', filter: {}, skip: 99996, '$db': 'learn' },
  serverInfo: {
    host: 'MacBook-Air-Ivan-2.local',
    port: 27017,
    version: '6.0.6',
    gitVersion: '26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
  },
  ok: 1
}

```

На выполнение запроса понадобилось 43 миллисекунды. По примерным подсчетам по секундомеру, на создание коллекции понадобилось сильно больше времени, около 50 секунд.

5. Создайте индекс для ключа value.

```

[learn> db.number.createIndex({value:1})
value_1

```


6. Получите информацию о всех индексах коллекции numbers.

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

Индекс поля value автоматически наименован как 'value_1'.

7. Выполните запрос 2.

```
learn> db.numbers.find({}).skip(99996)
[
  { _id: ObjectId("6477b9e20e5aba34a3f7705f"), value: 99996 },
  { _id: ObjectId("6477b9e20e5aba34a3f77060"), value: 99997 },
  { _id: ObjectId("6477b9e20e5aba34a3f77061"), value: 99998 },
  { _id: ObjectId("6477b9e20e5aba34a3f77062"), value: 99999 }
]
learn>
```

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

```
learn> db.numbers.explain('executionStats').find({}).skip(99996)
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'SKIP',
      skipAmount: 0,
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 37,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
```

На выполнение запроса с индексом, установленным на value, ушло 37 миллисекунд (на 6 миллисекунд меньше, чем без индекса).

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

Время выполнения запроса с индексом оказалось меньше, чем время запроса без индекса, с учетом того, что запрос был один и тот же и выполнялся на одной и той же коллекции. Таким образом, можно утверждать, что запрос с индексом более эффективен по времени (и, очевидно, разница будет более ощутимой с ростом количества данных в коллекции). Тем не менее, необходимо помнить об ограничениях использования индексов: их можно установить ограниченное количество, каждый из них использует оперативную память, и т.д.