ЛАБОРАТОРНАЯ РАБОТА №6.2

Работа с БД в СУБД MongoDB

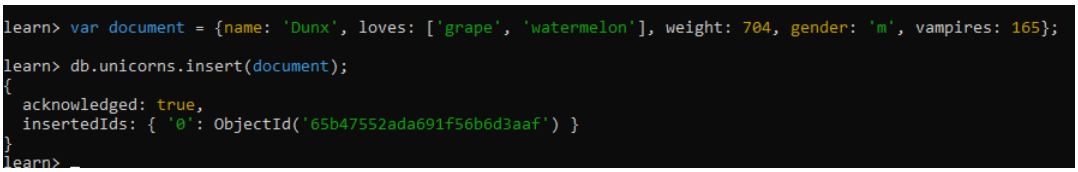
Цель: овладеть практическими навыками работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.

Програмное обеспечение: СУБД MongoDB

Практическое задание 2.1.1:

1,2)

3)



4)


learn> db.unicorns.find();
[
{
_id: ObjectId('65b47423ada691f56b6d3aa4'),
name: 'Horny',
loves: ['carrot', 'papaya'],
weight: 600,
gender: 'm',
vampires: 63
},
{
_id: ObjectId('65b47423ada691f56b6d3aa5'),
name: 'Aurora',
loves: ['carrot', 'grape'],
weight: 450,
gender: 'f',
vampires: 43
},
{
_id: ObjectId('65b47423ada691f56b6d3aa6'),
name: 'Unicrom',
loves: ['energon', 'redbull'],
learn>
gender: 'm',
vampires: 182
},
{
_id: ObjectId('65b47423ada691f56b6d3aa7'),
name: 'Roooooodles',
loves: ['apple'],
weight: 575,
gender: 'm',
vampires: 99
},
{
_id: ObjectId('65b47423ada691f56b6d3aa8'),
name: 'Solnara',
loves: ['apple', 'carrot', 'chocolate'],
weight: 550,
gender: 'f',
vampires: 80
},
{
_id: ObjectId('65b47424ada691f56b6d3aa9'),
name: 'Ayna',

Практическое задание 2.2.1:

1.Список самцов и самок единорогов, отсортированный по имени:

1) Для самцов (gender: ’m’):

```
learn> db.unicorns.find({gender: 'm'}).sort({name: 1});
[
  {
    _id: ObjectId('65b47552ada691f56b6d3aaf'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65b47423ada691f56b6d3aa4'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aaa'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aad'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aab'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65b47423ada691f56b6d3aa7'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
```

2) Для самок (gender: ’f’), ограниченный первыми тремя особями:

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3);
[
  {
    _id: ObjectId('65b47423ada691f56b6d3aa5'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aa9'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aac'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn>
```

2. Список самок, которые любят морковь (carrot):

1) Используя findOne (вернет только один документ):

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'});
{
  _id: ObjectId('65b47423ada691f56b6d3aa5'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```

2) Используя find и limit (вернет первую особь из списка):

```
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1);
[
  {
    _id: ObjectId('65b47423ada691f56b6d3aa5'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

3. Выборка только возраста (age) у всех документов, где имя (name) равно Tom:

```
learn> db.users.find({name: "Tom"}, {age: 1})
[
  { _id: ObjectId('65b47afb860ae342a5941971'), age: 28 },
  { _id: ObjectId('65b47afb860ae342a5941973'), age: 32 }
]
learn>
```

Обратная ситуация: нужно найти все параметры документа, кроме свойства age. В этом случае в качестве параметра указать 0:

```
learn> db.users.find({name: "Tom"}, {age: 0})
[
  {
    _id: ObjectId('65b47afb860ae342a5941971'),
    name: 'Tom',
    languages: [ 'english', 'spanish' ]
  },
  {
    _id: ObjectId('65b47afb860ae342a5941973'),
    name: 'Tom',
    languages: [ 'english', 'german' ]
  }
]
learn>
```

Альтернативно вместо 1 и 0 можно использовать true и false:

```
learn> db.users.find({name: "Tom"}, {age: true, _id: false})
[ { age: 28 }, { age: 32 } ]
learn>
```

Практическое задание 2.2.2:

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

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0}).sort({$natural: -1});
[
  {
    _id: ObjectId('65b47552ada691f56b6d3aaf'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aad'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aab'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aaa'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('65b47423ada691f56b6d3aa7'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('65b47423ada691f56b6d3aa6'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
  {
    _id: ObjectId('65b47423ada691f56b6d3aa4'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
]
```

Практическое задание 2.2.3: *Вывести список единорогов в обратном порядке добавления.*

```
learn> db.unicorns.find().sort({$natural: -1});
[
  {
    _id: ObjectId('65b47552ada691f56b6d3aaf'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aae'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aad'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aac'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aab'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65b47424ada691f56b6d3aaa'),
    name: 'Kenny',
```

Только первый язык из массива languages каждого пользователя:

```
learn> db.users.find({name: "Tom"}, {languages: {$slice: 1}});
[
  {
    _id: ObjectId('65b47afb860ae342a5941971'),
    name: 'Tom',
    age: 28,
    languages: [ 'english' ]
  },
  {
    _id: ObjectId('65b47afb860ae342a5941973'),
    name: 'Tom',
    age: 32,
    languages: [ 'english' ]
  }
]
learn>
```

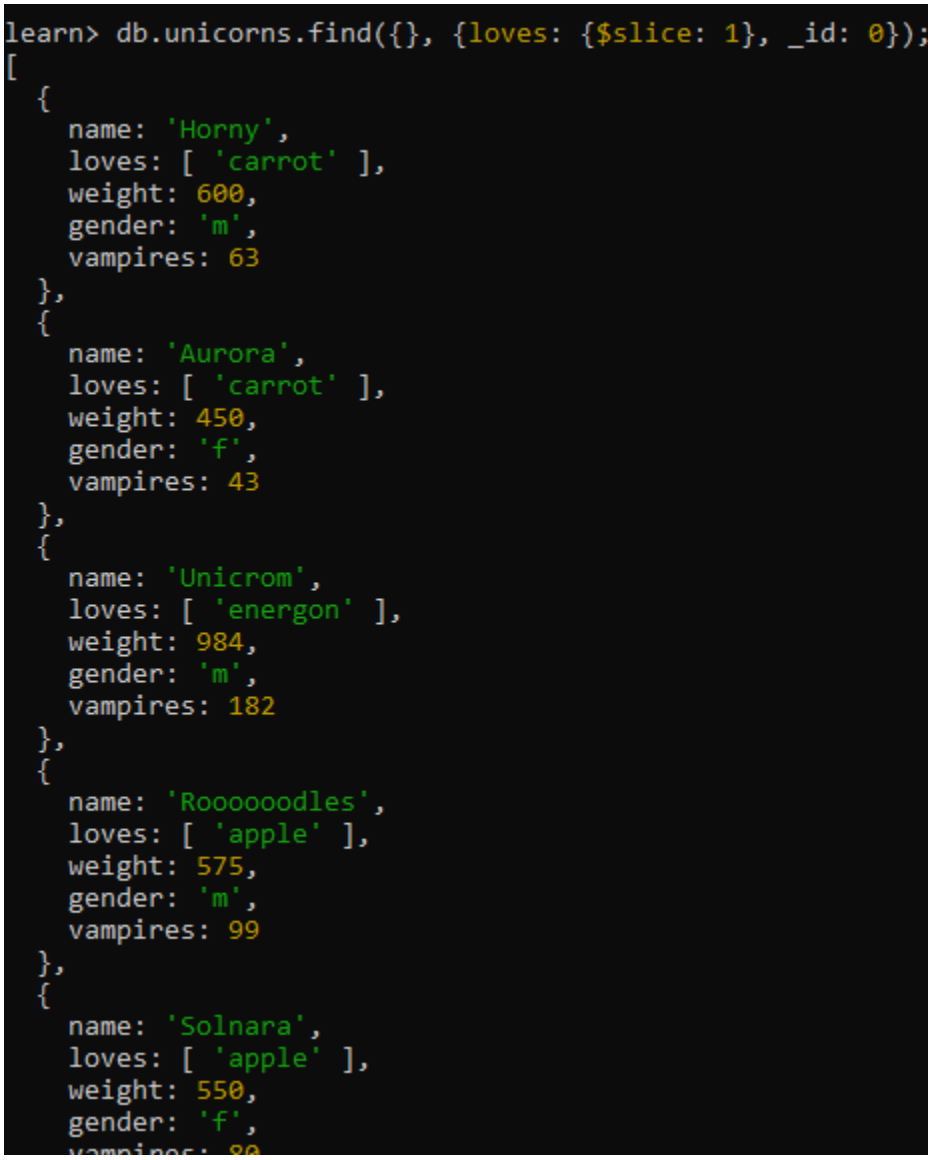
Только последний язык из массива languages:

```
learn> db.users.find({name: "Tom"}, {languages: {$slice: -1}});
[
  {
    _id: ObjectId('65b47afb860ae342a5941971'),
    name: 'Tom',
    age: 28,
    languages: [ 'spanish' ]
  },
  {
    _id: ObjectId('65b47afb860ae342a5941973'),
    name: 'Tom',
    age: 32,
    languages: [ 'german' ]
  }
]
learn>
```

Использование двух параметров (например, пропустить последний элемент и показать один элемент):

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:

```
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'
  }
]
learn>
```

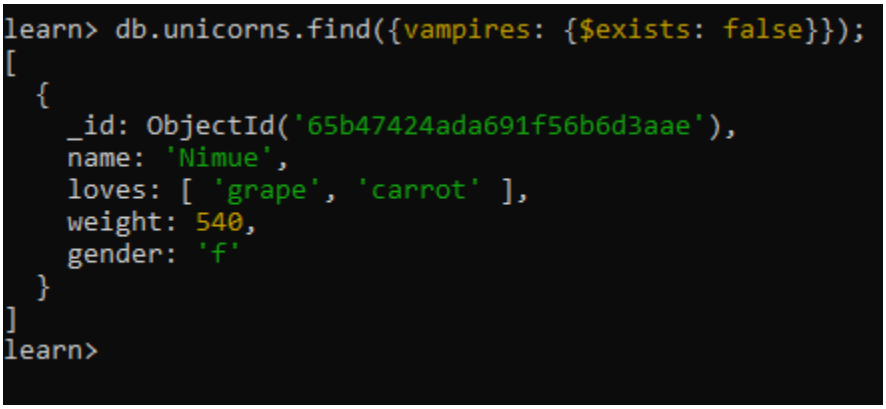
Практическое задание 2.3.2:

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", "lem
[
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
learn>
```

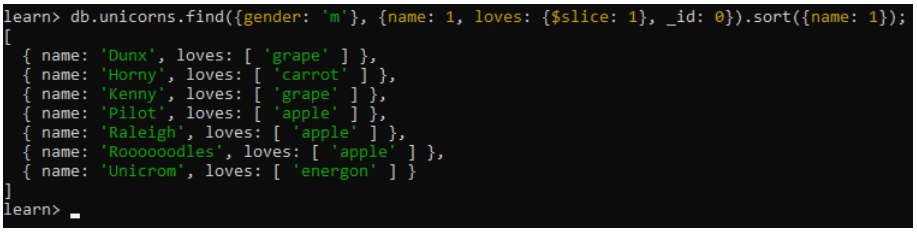
Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires:



Практическое задание 2.3.4:

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



1) Создание коллекции towns и добавление документов:

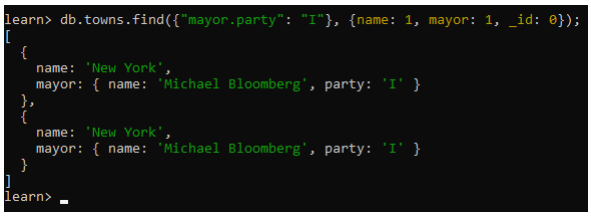
```
learn>

learn> db.towns.insert({
...     name: "Punxsutawney",
...     population: 6200,
...     last_census: ISODate("2008-01-31"),
...     famous_for: [],
...     mayor: {
...         name: "Jim Wehrle"
...     }
... });
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65b48296860ae342a5941977') }
}
learn>

learn> db.towns.insert({
...     name: "New York",
...     population: 22200000,
...     last_census: ISODate("2009-07-31"),
...     famous_for: ["statue of liberty", "food"],
...     mayor: {
...         name: "Michael Bloomberg",
...         party: "I"
...     }
... });
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65b48296860ae342a5941978') }
}
learn>

learn> db.towns.insert({
...     name: "Portland",
...     population: 528000,
...     last_census: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: {
...         name: "Sam Adams",
...         party: "D"
...     }
... });
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65b48296860ae342a5941979') }
}
learn>
```

2)Запрос на получение списка городов с независимыми мэрами (party="I"):



3) Запрос на получение списка городов с беспартийными мэрами (party отсутствует):

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

Практическое задание 3.1.2:

1) Функции для вывода списка самцов единорогов:

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

2) Курсор для списка самцов единорогов:

```
learn> var cursor = db.unicorns.find(maleUnicorns).sort({name: 1}).limit(2);

learn>
```

3) Вывод результатов с помощью forEach:

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

Практическое задание 3.2.1:

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

Практическое задание 3.2.2:

```
learn> db.unicorns.aggregate([
...     { $unwind: "$loves" },
...     { $group: { _id: "$loves", count: { $sum: 1 } } }
... ]);
[
  { _id: 'grape', count: 4 },
  { _id: 'papaya', count: 1 },
  { _id: 'lemon', count: 2 },
  { _id: 'redbull', count: 1 },
  { _id: 'apple', count: 5 },
  { _id: 'strawberry', count: 1 },
  { _id: 'carrot', count: 4 },
  { _id: 'watermelon', count: 3 },
  { _id: 'chocolate', count: 1 },
  { _id: 'energon', count: 1 },
  { _id: 'sugar', count: 1 }
]
learn>
```

Практическое задание 3.2.3:

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

Практическое задание 3.3.1: (тут не сработал метод, указаный в методичке: TypeError: db.unicorns.save is not a function)

```
learn> db.unicorns.updateOne(
...     { name: 'Barny' },
...     { $set: { loves: ['grape'], weight: 340, gender: 'm' } },
...     { upsert: true }
... );
{
  acknowledged: true,
  insertedId: ObjectId('65b48a61a81557b7fa9a68e2'),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
learn>
```

```
{
  _id: ObjectId('65b47424ada691f56b6d3aad'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon' ],
  weight: 650,
  gender: 'm',
  vampires: 54
},
{
  _id: ObjectId('65b47424ada691f56b6d3aae'),
  name: 'Nimue',
  loves: [ 'grape', 'carrot' ],
  weight: 540,
  gender: 'f'
},
{
  _id: ObjectId('65b47552ada691f56b6d3aaf'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
},
{
  _id: ObjectId('65b48a61a81557b7fa9a68e2'),
  name: 'Barny',
  gender: 'm',
  loves: [ 'grape' ],
  weight: 340
```

Практическое задание 3.3.2:

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

```
learn> db.unicorns.find({ name: "Ayna" });
[
  {
    _id: ObjectId('65b47424ada691f56b6d3aa9'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
learn>
```

Практическое задание 3.3.3:

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

```
learn> db.unicorns.find({ name: "Raleigh" });
[
  {
    _id: ObjectId('65b47424ada691f56b6d3aab'),
    name: 'Raleigh',
    loves: [ 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
learn>
```

Практическое задание 3.3.4:

```
learn> db.unicorns.updateMany(
...     { gender: "m" },
...     { $inc: { vampires: 5 } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 9,
  modifiedCount: 9,
  upsertedCount: 0
}
learn>
```

Практическое задание 3.3.5:

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

```
learn> db.towns.find({ name: "Portland" });
[
  {
    _id: ObjectId('65b4821e860ae342a5941976'),
    name: 'Portland',
    population: 528000,
    last_census: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  },
  {
    _id: ObjectId('65b48296860ae342a5941979'),
    name: 'Portland',
    population: 528000,
    last_census: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>
```

Практическое задание 3.3.6:

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

```
learn> db.unicorns.find({ name: "Pilot" });
[
  {
    _id: ObjectId('65b47424ada691f56b6d3aad'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
learn>
```

Практическое задание 3.3.7:

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

```
learn> db.unicorns.find({ name: "Aurora" });
[
  {
    _id: ObjectId('65b47423ada691f56b6d3aa5'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn>
```

Практическое задание 3.4.1: *Создание коллекции towns и добавление документов:*

```
learn> db.towns.insertMany([
...   {name: "Punxsutawney", population: 6200, last_census: ISODate("2008-01-31"), famous_for: ["phil the groundhog"], mayor: {name: "Jim Wehrle"}},
...   {name: "New York", population: 22200000, last_census: ISODate("2009-07-31"), famous_for: ["statue of liberty", "food"], mayor: {name: "Michael Bloomberg", party: "I"}},
...   {name: "Portland", population: 528000, last_census: ISODate("2009-07-20"), famous_for: ["beer", "food"], mayor: {name: "Sam Adams", party: "D"}}
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65b498e6860ae342a594197e'),
    '1': ObjectId('65b498e6860ae342a594197f'),
    '2': ObjectId('65b498e6860ae342a5941980')
  }
}
learn>
```

Удаление документов с беспартийными мэрами:

```
learn> db.towns.remove({"mayor.party": {$exists: false}});
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 5 }
learn>
```

Проверка содержимого коллекции towns:

```
learn> db.towns.find();
[
  {
    _id: ObjectId('65b4821e860ae342a5941975'),
    name: 'New York',
    population: 22200000,
    last_census: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('65b48296860ae342a5941978'),
    name: 'New York',
    population: 22200000,
    last_census: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('65b48296860ae342a5941979'),
    name: 'Portland',
    population: 528000,
    last_census: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  },
  {
    _id: ObjectId('65b48fb6860ae342a594197c'),
    name: 'New York',
    population: 22200000,
    last_census: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('65b48fb6860ae342a594197d'),
    name: 'Portland',
    population: 528000,
    last_census: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  },
  {
    _id: ObjectId('65b498e6860ae342a594197f'),
    name: 'New York',
    population: 22200000,
    last_census: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'statue of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('65b498e6860ae342a5941980'),
    name: 'Portland',
    population: 528000,
    last_census: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

Очистка коллекции towns:

```
learn> db.towns.remove({});
{ acknowledged: true, deletedCount: 7 }
learn>
```

Просмотр списка доступных коллекций:

```
learn> db.getCollectionNames();
[ 'towns', 'users', 'unicorns' ]
learn>
```

Практическое задание 4.1.1: *Создание коллекции зон обитания единорогов:*

```
learn> db.habitats.insertMany([
...   { _id: "les", name: "Зачарованный Лес", description: "Таинственное и магическое место." },
...   { _id: "gora", name: "Кристальная Гора", description: "Гора, покрытая кристаллами." }
... ]);
{ acknowledged: true, insertedIds: { '0': 'les', '1': 'gora' } }
learn>
```

Обновление единорогов с ссылкой на зону обитания:

```
learn> db.unicorns.updateOne(
...   { name: 'Horny' },
...   { $set: { habitat: { $ref: "habitats", $id: "les" } } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateOne(
...   { name: 'Aurora' },
...   { $set: { habitat: { $ref: "habitats", $id: "les" } } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>
```

Проверка содержания коллекции unicorns:

```
learn> db.unicorns.find();
[
  {
    _id: ObjectId('65b47423ada691f56b6d3aa4'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    habitat: DBRef('habitats', 'les')
  },
  {
    _id: ObjectId('65b47423ada691f56b6d3aa5'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    habitat: DBRef('habitats', 'les')
  },
  {
    _id: ObjectId('65b47423ada691f56b6d3aa6'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 187
  },
  {
    _id: ObjectId('65b47423ada691f56b6d3aa7'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 104
  },
  {
    _id: ObjectId('65b47423ada691f56b6d3aa8'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550
```

Практическое задание 4.2.1: *Создание уникального индекса:*

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

Проверка содержимого коллекции unicorns:

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

Вообще, создать ключ уникальный индекс не получалось, поэтому, пришлось переименовать одного единорога:

```
learn> db.unicorns.deleteOne({ name: 'Barny' });
{ acknowledged: true, deletedCount: 1 }
learn>
```

Практическое задание 4.3.1: *Получение информации об индексах в коллекции unicorns:*

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

Получение списка индексов и удаление всех, кроме id:

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

Попытка удаления индекса для id:

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

Практическое задание 4.4.1: *Создание коллекции numbers и добавление 100000 документов:*

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

Выбор последних четырех документов и анализ плана выполнения запроса:

```
learn> db.numbers.explain("executionStats").find({}).sort({value: -1}).limit(4);
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: 'D63FBA81',
    planCacheKey: 'D63FBA81',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'SORT',
      sortPattern: { value: -1 },
      memLimit: 104857600,
      limitAmount: 4,
      type: 'simple',
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 59,
    totalKeysExamined: 0,
    totalDocsExamined: 100000,
    executionStages: {
      stage: 'SORT',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 100006,
      advanced: 4,
      needTime: 100001,
      needYield: 0,
      saveState: 100,
      restoreState: 100,
      isEOF: 1,
      sortPattern: { value: -1 },
      memLimit: 104857600,
      limitAmount: 4,
      type: 'simple',
      totalDataSizeSorted: 260,
      usedDisk: false,
      spills: 0,
      spilledDataStorageSize: 0,
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 100000,
        executionTimeMillisEstimate: 0,
        works: 100001,
        advanced: 100000,
        needTime: 0,
        needYield: 0,
        saveState: 100,
        restoreState: 100,
        isEOF: 1,
        direction: 'forward',
        docsExamined: 100000
      }
    }
```

executionTimeMillis: 59, Создание индекса для ключа value:

```
learn> db.numbers.createIndex({value: 1});
value_1
learn>
```

Получение информации об индексах коллекции numbers:

```
learn> db.numbers.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
learn>
```

Выполнение запроса с установленным индексом:

До создания индекса: executionTimeMillis: 59 миллисекунд

totalDocsExamined: 100000 документов

План выполнения запроса показывает, что использовался COLLSCAN (полное сканирование коллекции).

После создания индекса:

executionTimeMillis: 1 миллисекунда

totalDocsExamined: 4 документа

План выполнения запроса показывает, что использовался IXSCAN (сканирование индекса).