

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО

ОТЧЕТ

по Лабораторной работе № 5

«процедуры, функции, триггеры в PostgreSQL»

по дисциплине «Проектирование и реализация баз данных»

Обучающиеся Захматов Юрий Дмитриевич

Факультет прикладной информатики

Группа К3241

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2023

Преподаватель Говорова Марина Михайловна

Санкт-Петербург 2024/2025

Цель работы: овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

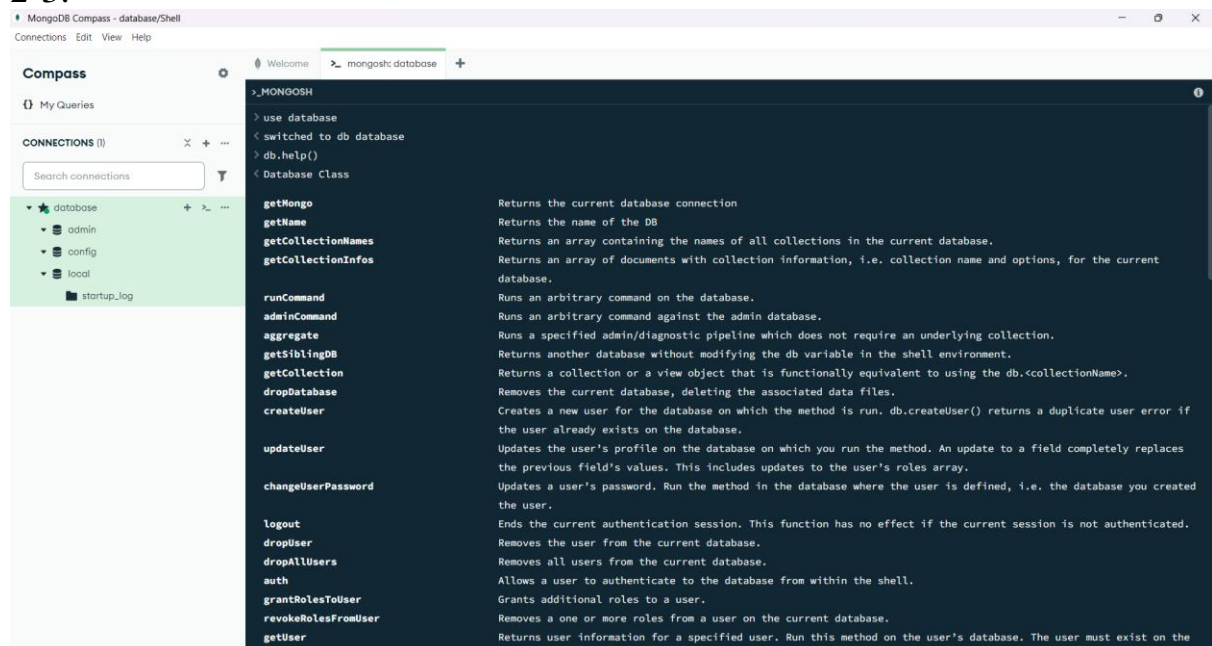
Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, pgadmin 4, SQL Shell (psql).

Практическое задание 1:

1. Установите MongoDB для обеих типов систем (32/64 бита).
2. Проверьте работоспособность системы запуском клиента mongo.
3. Выполните методы:
 - a. db.help()
 - b. db.help
 - c. db.stats()
4. Создайте БД learn.
5. Получите список доступных БД.
6. Создайте коллекцию unicorns, вставив в нее документ {name: 'Aurora', gender: 'f', weight: 450}.
7. Просмотрите список текущих коллекций.
8. Переименуйте коллекцию unicorns.
9. Просмотрите статистику коллекции.
10. Удалите коллекцию.
11. Удалите БД learn.

2-3:



4-5: создал database вместо learn

```
> show databases
< admin      40.00 KiB
   config    60.00 KiB
   database   8.00 KiB
   local     40.00 KiB
```

6:

```
> db.unicorns.insert({name: 'Aurora', gender: 'f', weight: 450})
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6835d8981851f02869d88fa5')
  }
}
```

7:

```
> show collections
< learn
   unicorns
> db.unicorns.renameCollection("unicorns_renamed")
```

8:

```
> db.unicorns.renameCollection("unicorns_renamed")
< { ok: 1 }
```

9:

```
> db.unicorns_renamed.stats
< [Function: stats] AsyncFunction {
  apiVersions: [ 0, 0 ],
  returnsPromise: true,
  serverVersions: [ '0.0.0', '999.999.999' ],
  topologies: [ 'ReplSet', 'Sharded', 'LoadBalanced', 'Standalone' ],
  returnType: { type: 'unknown', attributes: {} },
  deprecated: false,
  platforms: [ 'Compass', 'Browser', 'CLI' ],
  isDirectShellCommand: false,
  acceptsRawInput: false,
  shellCommandCompleter: undefined,
  help: [Function (anonymous)] Help
}
```

10-11:

```
> db.unicorns_renamed.drop()
< true
> db.dropDatabase()
< { ok: 1, dropped: 'database' }
```

Практическое задание 2.1.1

1. Создайте базу данных *learn*.
2. 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: 'Rooooooodles', 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'});

```

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

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

```

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

1-2:

```

> use learn
< switched to db 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: 'Rooooooodles', 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'});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
 acknowledged: true,
 insertedIds: {
 '0': ObjectId('6835dbbcd4ebc5066077c9ac')
 }
}

```

3:

```

> 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.insert(document)
< {
 acknowledged: true,
 insertedIds: {
 '0': ObjectId('6835dc2dd4ebc5066077c9ad')
 }
}
> db.unicorns.find()

```

4:

```
> db.unicorns.find()
< {
 _id: ObjectId('6835dbbcd4ebc5066077c9a2'),
 name: 'Horny',
 loves: [
 'carrot',
 'papaya'
],
 weight: 600,
 gender: 'm',
 vampires: 63
}
{
 _id: ObjectId('6835dbbcd4ebc5066077c9a3'),
 name: 'Aurora',
 loves: [
 'carrot',
 'grape'
],
 weight: 450,
 gender: 'f',
 vampires: 43
}
{
 _id: ObjectId('6835dbbcd4ebc5066077c9a4'),
 name: 'Unicrom',
 loves: [
 'energon',
 'redbull'
```

## Практическое задание 2.1.2:

1. Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.
2. Найдите всех самок, которые любят carrot. Ограничьте этот список первой особью с помощью функций *findOne* и *limit*.

1:

```
db.unicorns.find({gender: 'm'}).sort({name: 1})
```

```
db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
```

```

}
> db.unicorns.find({gender: 'm'}).sort({name: 1})
< {
 _id: ObjectId('6835dc2dd4ebc5066077c9ad'),
 name: 'Dunx',
 loves: [
 'grape',
 'watermelon'
],
 weight: 704,
 gender: 'm',
 vampires: 165
}
{
 _id: ObjectId('6835dbbcd4ebc5066077c9a2'),
 name: 'Horny',
 loves: [
 'carrot',
 'papaya'
],
 weight: 600
}
> db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
< {
 _id: ObjectId('6835dbbcd4ebc5066077c9a3'),
 name: 'Aurora',
 loves: [
 'carrot',
 'grape'
],
 weight: 450,
 gender: 'f',
 vampires: 43
}
{
 _id: ObjectId('6835dbbcd4ebc5066077c9a7'),
 name: 'Ayna',
 loves: [
 'watermelon',
 'grape'
],
 weight: 500,
 gender: 'f',
 vampires: 40
}

```

```
2: db.unicorns.findOne({loves: "carrot"})
 db.unicorns.find({loves: "carrot"}).limit(1)
```

```
> db.unicorns.findOne({loves: "carrot"})
< {
 _id: ObjectId('6835dbbcd4ebc5066077c9a2'),
 name: 'Horny',
 loves: [
 'carrot',
 'papaya'
],
 weight: 600,
 gender: 'm',
 vampires: 63
}
> db.unicorns.find({loves: "carrot"}).limit(1)
< {
 _id: ObjectId('6835dbbcd4ebc5066077c9a2'),
 name: 'Horny',
 loves: [
 'carrot',
 'papaya'
],
 weight: 600,
 gender: 'm',
 vampires: 63
}
learn >
```

### Практическое задание 2.2.1:

Модифицируйте запрос для вывода списков самцов единорогов, исключив из результата информацию о предпочтениях и поле (видимо *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('6835dc2dd4ebc5066077c9ad'),
 name: 'Dunx',
 weight: 704,
 gender: 'm'
}
```

### Практическое задание 2.2.3

*Вывести список единорогов в обратном порядке добавления.*

**db.unicorns.find().sort({\$natural: -1})**

```
> db.unicorns.find().sort({$natural: -1})
< {
 _id: ObjectId('6835dc2dd4ebc5066077c9ad'),
 name: 'Dunx',
 loves: [
 'grape',
 'watermelon'
],
 weight: 704,
 gender: 'm',
 vampires: 165
}
```

### Практическое задание 2.2.4

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

**db.unicorns.find({}, {\_id: 0, name: 1, loves: {\$slice: 1}})**

```
> db.unicorns.find({}, {_id: 0, name: 1, loves: {$slice: 1}})
< {
 name: 'Horny',
 loves: [
 'carrot'
]
}
{
 name: 'Aurora',
 loves: [
 'carrot'
]
}
```

### Практическое задание 2.3.1

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

```
db.unicorns.find({weight: {$gt: 500, $lt: 700}}, {_id: 0})
```

```
> db.unicorns.find({weight: {$gt: 500, $lt: 700}}, {_id: 0})
< [{
 name: 'Horny',
 loves: [
 'carrot',
 'papaya'
],
 weight: 600,
 gender: 'm',
 vampires: 63
}]
```

### Практическое задание 2.3.2

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

```
db.unicorns.find({weight: {$gt: 500, $lt: 700}, loves: {$all : ["grape","lemon"]}}, {_id: 0})
```

```

 }
 }
 > db.unicorns.find({weight: {$gt: 500, $lt: 700}, loves: {$all : ["grape","lemon"]}}, {_id: 0})
< {
 name: 'Kenny',
 loves: [
 'grape',
 'lemon'
],
 weight: 690,
 gender: 'm',
 vampires: 39
}
learn>

```

### Практическое задание 2.3.3

Найти всех единорогов, не имеющих ключ *vampires*.

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

```

> db.unicorns.find({vampires: {$exists: false}})
< {
 _id: ObjectId('6835dbbcd4ebc5066077c9ac'),
 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.

```
db.unicorns.find({gender: 'm'}, {_id: 0, name: 1, loves: {$slice: 1}}).sort({name: 1})
```

```

 }
 }
 > db.unicorns.find({gender: 'm'}, {_id: 0, name: 1, loves: {$slice: 1}}).sort({name: 1})
< {
 name: 'Dunx',
 loves: [
 'grape'
]
}

```

### Практическое задание 3.1.1

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

1: `db.towns.find({"mayor.party" : "I"}, {name: 1, mayor: 1})`

```
> db.towns.find({"mayor.party" : "I"}, {name: 1, mayor: 1})
< {
 _id: ObjectId('6835f06ed4ebc5066077c9b0'),
 name: 'New York',
 mayor: {
 name: 'Michael Bloomberg',
 party: 'I'
 }
}
```

2: `db.towns.find({"mayor.party" : {$exists: false}}, {name: 1, mayor: 1})`

```
> db.towns.find({"mayor.party" : {$exists: false}}, {name: 1, mayor: 1})
< {
 _id: ObjectId('6835f055d4ebc5066077c9af'),
 name: 'Punxsutawney ',
 mayor: {
 name: 'Jim Wehrle'
 }
}
```

### Практическое задание 3.1.2

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' }, { _id: 0 }).sort({ name: 1 }).limit(2); null;
cursor.forEach(function(unicorn) { print(unicorn.name); });
```

```
> var cursor = db.unicorns.find({ gender: 'm' }, { _id: 0 }).sort({ name: 1 }).limit(2); null;
< null
> cursor.forEach(function(unicorn) {print(unicorn.name);});
< Dunx
< Horny
learn> +
```

### Практическое задание 3.2.1

*Вывести количество самок единорогов весом от полутонны до 600 кг.*

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

### Практическое задание 3.2.2

*Вывести список предпочтений.*

**db.unicorns.distinct("loves")**

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

### Практическое задание 3.2.3

*Посчитать количество особей единорогов обоих полов.*

**db.unicorns.find({gender: {\$in: ['m','f']}}).count()**

```
> db.unicorns.find({gender: {$in: ['m','f']}}).count()
< 12
```

### Практическое задание 3.3.1

1. *Выполнить команду:*

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

## 2. Проверить содержимое коллекции *unicorns*.

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

Немного погуглив, я узнал что эта функция был вырезана из MongoDB.

2 Answers

Sorted by: Highest score (default)



.save() has been deprecated instead of that .save() you can use .insertOne() or .insertMany() or .updateOne({upsert:true})

8

You can write your code like below by replacing .save() with .insertOne()

### Практическое задание 3.3.2

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

2. Проверить содержимое коллекции *unicorns*.

```
> db.unicorns.update({name: "Ayna" }, {$set: {weight: 800, vampires: 51}}, { upsert: true });
{
 acknowledged: true,
 insertedId: null,
 matchedCount: 1,
 modifiedCount: 1,
 upsertedCount: 0
}
{
 _id: ObjectId('6835f6a3d4ebc5066077c9c3'),
 name: 'Ayna',
 loves: [
 'strawberry',
 'lemon'
],
 weight: 800,
 gender: 'f',
 vampires: 51
}
```

### Практическое задание 3.3.3

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

Проверить содержимое коллекции *unicorns*.

```

> db.unicorns.update({name: "Raleigh" }, {$set: {loves: "redbull"}},{ upsert: true });
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 1,
 modifiedCount: 1,
 upsertedCount: 0
}
> db.unicorns.find({name: "Raleigh"})
< {
 _id: ObjectId('6835f6a3d4ebc5066077c9c5'),
 name: 'Raleigh',
 loves: 'redbull',
 weight: 421,
 gender: 'm',
 vampires: 2
}

```

### Практическое задание 3.3.4

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

Проверить содержимое коллекции *unicorns*.

```

> db.unicorns.update({gender: "m" }, {$inc: {vampires: 5}},{ upsert: true, multi: true });
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 7,
 modifiedCount: 7,
 upsertedCount: 0
}

```

```

> db.unicorns.find({gender: "m"})
< {
 _id: ObjectId('6835f6a3d4ebc5066077c9be'),
 name: 'Horny',
 loves: [
 'carrot',
 'papaya'
],
 weight: 600,
 gender: 'm',
 vampires: 68
}
{
 _id: ObjectId('6835f6a3d4ebc5066077c9c0'),
 name: 'Unicrom',
 loves: [
 'energon',
 'redbull'
],
 weight: 984,
 gender: 'm',
 vampires: 187
}

```

### Практическое задание 3.3.5

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

```

> db.towns.update({name: "Portland"}, {$set: {"mayor.party": "I"}})
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 1,
 modifiedCount: 1,
 upsertedCount: 0
}

```



```
> db.towns.find({name: "Portland"})
< {
 _id: ObjectId('6835f08cd4ebc5066077c9b1'),
 name: 'Portland',
 populatiuon: 528000,
 last_sensus: 2009-07-20T00:00:00.000Z,
 famous_for: [
 'beer',
 'food'
],
 mayor: {
 name: 'Sam Adams',
 party: 'I'
 }
}
```

### Практическое задание 3.3.6

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

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

```
> db.unicorns.find({name: "Pilot"})
< {
 _id: ObjectId('6835f6a3d4ebc5066077c9c7'),
 name: 'Pilot',
 loves: [
 'apple',
 'watermelon',
 'chocolate'
],
 weight: 650,
 gender: 'm',
 vampires: 59
}
```

### Практическое задание 3.3.7

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

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

> db.unicorns.find({name: "Aurora"})
< {
 _id: ObjectId('6835f6a3d4ebc5066077c9bf'),
 name: 'Aurora',
 loves: [
 'carrot',
 'grape',
 'sugar',
 'lemons'
],
 weight: 450,
 gender: 'f',
 vampires: 43
}
```

### Практическое задание 3.4.1

1. Удалите документы с беспартийными мэрами.
2. Проверьте содержание коллекции.
3. Очистите коллекцию.
4. Просмотрите список доступных коллекций.

```
> db.towns.remove({"mayor.party": "I"})
< DeprecationWarning: Collection.remove() is deprecated
< {
 acknowledged: true,
 deletedCount: 1
}
```

```
> db.towns.find()
< {
 _id: ObjectId('6836ba2cb4526f7e82690f24'),
 name: 'Punxsutawney ',
 popujatiuon: 6200,
 last_sensus: 2008-01-31T00:00:00.000Z,
 famous_for: [
 'phil the groundhog'
],
 mayor: {
 name: 'Jim Wehrle'
 }
}
{
 _id: ObjectId('6836ba53b4526f7e82690f26'),
 name: 'Portland',
 popujatiuon: 528000,
 last_sensus: 2009-07-20T00:00:00.000Z,
 famous_for: [
 'beer',
 'food'
],
 mayor: {
 name: 'Sam Adams',
 party: 'D'
 }
}
learn>
```

```
> db.towns.remove({})
< {
 acknowledged: true,
 deletedCount: 2
}
```

```
> show collections
< towns
 unicorns
learn> |
```

### Практическое задание 4.1.1

1. Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.
2. Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания.
3. Проверьте содержание коллекции единорогов `unicorns`:

```
db.habitats.insertMany([{_id: "forest",fullName: "Enchanted Forest"},{_id: "mountains", fullName: "Crystal Mountains"}, {_id: "meadows", fullName: "Rainbow Meadows"}, {_id: "clouds", fullName: "Floating Cloud Islands"}])
```

```
unicorns
> db.habitats.insertMany([{_id: "forest",fullName: "Enchanted Forest"},{_id: "mountains", fullName: "Crystal Mountains"}, {_id: "meadows", f
< {
 acknowledged: true,
 insertedIds: {
 '0': 'forest',
 '1': 'mountains',
 '2': 'meadows',
 '3': 'clouds'
 }
}
```

```
> db.unicorns.updateMany({name: {$in:["Horny", "Aurora", "Unicrom"]}},{$set:{ habitatId: "forest"}})
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 3,
 modifiedCount: 3,
 upsertedCount: 0
}
```

```
> db.unicorns.updateMany({name: {$in: ["Rooodooles", "Solnara", "Ayna"]}},{$set: {habitatId: "mountains"}})
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 3,
 modifiedCount: 3,
 upsertedCount: 0
}
```

```
> db.unicorns.updateMany({name: {$in: ["Kenny", "Raleigh", "Leia"]}},{$set: {habitatId: "meadows"}})
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 3,
 modifiedCount: 3,
 upsertedCount: 0
}
```

```
> db.unicorns.updateMany({name: {$in: ["Pilot", "Nimue", "Dunx"]}},{$set: { habitatId: "clouds"}})
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 3,
 modifiedCount: 3,
 upsertedCount: 0
}
```

```
> db.unicorns.find().pretty()
< {
 _id: ObjectId('6835f6a3d4ebc5066077c9be'),
 name: 'Horny',
 loves: [
 'carrot',
 'papaya'
],
 weight: 600,
 gender: 'm',
 vampires: 68,
 habitatId: 'forest'
}
{
 _id: ObjectId('6835f6a3d4ebc5066077c9bf'),
 name: 'Aurora',
 loves: [
 'carrot',
 'grape',
 'sugar',
 'lemons'
],
 weight: 450,
 gender: 'f',
 vampires: 43,
 habitatId: 'forest'
}
```



### Практическое задание 4.2.1

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

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

Можно

### Практическое задание 4.3.1

1. Получите информацию о всех индексах коллекции `unicorns`.
2. Удалите все индексы, кроме индекса для идентификатора.
3. Попробуйте удалить индекс для идентификатора.

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

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

```
> db.unicorns.dropIndex("_id_")
✖ ▸ MongoServerError[InvalidOptions]: cannot drop _id index
learn> |
```

### 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.
3. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра `executionTimeMillis`)
4. Создайте индекс для ключа `value`.
5. Получите информацию о всех индексах коллекции `numbers`.

6. Выполните запрос 2.
7. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?
8. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

```
> for(i = 0; i < 100000; i++) { db.numbers.insert({value: i}) }
< {
 acknowledged: true,
 insertedIds: {
 '0': ObjectId('6836cfd3b4526f7e826a95c6')
 }
}
```

```
> db.numbers.find().sort({_id: -1}).limit(4)
< {
 _id: ObjectId('6836cfd3b4526f7e826a95c6'),
 value: 99999
}
{
 _id: ObjectId('6836cfd3b4526f7e826a95c5'),
 value: 99998
}
{
 _id: ObjectId('6836cfd3b4526f7e826a95c4'),
 value: 99997
}
{
 _id: ObjectId('6836cfd3b4526f7e826a95c3'),
 value: 99996
}
```

```
nReturned: 1,
executionTimeMillis: 72,
totalKeysExamined: 0,
totalDocsExamined: 100000,
```

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

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

```
nReturned: 1,
executionTimeMillisEstimate: 23,
works: 2,
```

**Вывод:** с индексом быстрее.