

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 «Проектирование и реализация баз данных»

Обучающийся Никульшин Егор Сергеевич

Факультет прикладной информатики

Группа K3241

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии

Преподаватель Говорова Марина Михайловна

Санкт-Петербург 2024/2025

Лабораторная работа 6.1

Практическое задание:

1. Установите MongoDB для обеих типов систем (32/64 бита).
2. Проверьте работоспособность системы запуском клиента mongo.
3. Выполните методы:
 - db.help()
 - db.help
 - 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.

```
> db.help()
< Database Class

getMongo           Returns the current database connection
getName            Returns the name of the DB
getCollectionNames Returns an array containing the names of all collections in the current database.
getCollectionInfos  Returns an array of documents with collection information, i.e. collection name and options, for the
                    current database.

runCommand          Runs an arbitrary command on the database.
adminCommand        Runs an arbitrary command against the admin database.
aggregate           Runs a specified admin/diagnostic pipeline which does not require an underlying collection.
getSiblingDB        Returns another database without modifying the db variable in the shell environment.
getCollection        Returns a collection or a view object that is functionally equivalent to using the db.<collectionName>.
dropDatabase        Removes the current database, deleting the associated data files.
createUser           Creates a new user for the database on which the method is run. db.createUser() returns a duplicate
                    user error if the user already exists on the database.
updateUser           Updates the user's profile on the database on which you run the method. An update to a field completely
                    replaces the previous field's values. This includes updates to the user's roles array.
changeUserPassword  Updates a user's password. Run the method in the database where the user is defined, i.e. the database
```

Рисунок 4

```
> use learn
< switched to db learn
```

Рисунок 5

```
> show databases
< admin      40.00 KiB
  config    108.00 KiB
  learn      4.83 MiB
  local     40.00 KiB
```

Рисунок 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')
  }
}

```

Рисунок 7

```

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

```

Рисунок 7

```

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

```

Рисунок 8

```

> 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
}
> show databases

```

Рисунок 9

```

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

```

Рисунок 10-11

Лабораторная работа 6.2

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: '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'});

```

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*.

```

>_MONGOSH

> 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('6836c159f8bf637f5e8f7c39')
  }
}

```

```

> doc= {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(doc)
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6836c20ef8bf637f5e8f7c3a')
  }
}

```

```

> db.unicorns.find({ name: 'Dunx' }).pretty()
< {
  _id: ObjectId('6836c20ef8bf637f5e8f7c3a'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}

```

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.

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

```
}
> db.unicorns.find({gender: "m"}).sort({name: 1})
< {
  _id: ObjectId('6836c20ef8bf637f5e8f7c3a'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('6836c159f8bf637f5e8f7c2f'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

```

    }
  > db.unicorns.find({gender: "f"}).sort({name: 1}).limit(3)
  < {
    _id: ObjectId('6836c159f8bf637f5e8f7c30'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('6836c159f8bf637f5e8f7c34'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 43
  }

```

1 Рисулки

```

> db.unicorns.findOne({
  gender: "f",
  loves: "carrot"
})
< {
  _id: ObjectId('6836c159f8bf637f5e8f7c30'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn> |

```

```
> db.unicorns.find({
  gender: "f",
  loves: "carrot"
}).limit(1)
< {
  _id: ObjectId('6836c159f8bf637f5e8f7c30'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn >
```

Для 2 задания рисунки

Практическое задание 2.2.2:

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

Если нужно отсортировать ограниченную коллекцию, то можно воспользоваться параметром `$natural`. Этот параметр позволяет задать сортировку: документы передаются в том порядке, в каком они были добавлены в коллекцию, либо в обратном порядке.

Например, отобразить последние пять документов:

```
> db.users.find().sort({ $natural: -1 }).limit(5)
```



```

> db.unicorns.find(
  { gender: "m" }, { loves: 0, gender: 0, _id: 0 } ).sort({ name: 1 })
< {
  name: 'Dunx',
  weight: 704,
  vampires: 165
}
{
  name: 'Horny',
  weight: 600,
  vampires: 63
}

```

Рисунок 1

Практическое задание 2.2.3:

1. Вывести список единорогов в обратном порядке добавления.

```

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

```

Рисунок 1

Практическое задание 2.1.4:

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

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

Рисунок 1

Практическое задание 2.3.1:

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

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

1 Рисунок

Практическое задание 2.3.2

1. 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})
< {
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}

```

Рисунок 1

Практическое задание 2.3.3

1. *Найти всех единорогов, не имеющих ключ vampires.*

```

}
> db.unicorns.find({vampires: {$exists: false}})
< {
  _id: ObjectId('6836c159f8bf637f5e8f7c39'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
learn>

```

1 Рисунок

Практическое задание 2.3.4

1. *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})
< {
  name: 'Dunx',
  loves: [
    'grape'
  ]
}
{
  name: 'Horny',
  loves: [
    'carrot'
  ]
}
{
  name: 'Kenny',
  loves: [
    'grape'
  ]
}
{
  name: 'Pilot',
  loves: [
    'carrot'
  ]
}

```

Рисунок 1

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.

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

```
> db.towns.insertMany([
  {
    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"
    }
  },
])
```

Рисунок 1

```
> db.towns.find({"mayor.party" : "I"}, {name: 1, mayor: 1})
< {
  _id: ObjectId('68373de3e192ff0624961061'),
  name: 'New York',
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
learn>
```

Рисунок 2

```

> db.towns.find({"mayor.party" : {$exists: false}}, {name: 1, mayor: 1})
< {
  _id: ObjectId('68373de3e192ff0624961060'),
  name: 'Punxsutawney',
  mayor: {
    name: 'Jim Wehrle'
  }
}
learn>

```

Рисунок 3

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)});
< Dunx
< Horny
learn> |

```

Рисунок 1-3

Практическое задание 3.2.1

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

```

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

```

Рисунок 1

Практическое задание 3.2.2

1. Вывести список предпочтений.

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

Рисунок 1

Практическое задание 3.2.3

1. Посчитать количество особей единорогов обоих полов.

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

Рисунок 1

Практическое задание 3.3.1

1. Выполнить команду:

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

2. Проверить содержимое коллекции *unicorns*.

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

Рисунок 1

(Эта функция была удалена из версии этой в MongoDB)

Практическое задание 3.3.2

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

2. Проверить содержимое коллекции *unicorns*.

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

1 Рисунок

```
> db.unicorns.find(
  { name: "Ayna" },
  { _id: 0 }).pretty()
< {
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 800,
  gender: 'f',
  vampires: 51
}
learn> |
```

2 Рисунок

Практическое задание 3.3.3

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

```
> db.unicorns.update(
  { name: " Raleigh", gender: "f" },
  {
    $set: {
      loves: "redbull"
    }, {upsert: true});
< {
  acknowledged: true,
  insertedId: ObjectId('6837459528b7ccd29ad85a5c'),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
```

Рисунок 1

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.find(
  { name: "Raleigh" })
< {
  _id: ObjectId('6836c159f8bf637f5e8f7c36'),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
```

Рисунок 2

Практическое задание 3.3.4

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.
2. 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
}
```

Рисунок 1

```
> 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
}
```

Рисунок 2

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" }  
  }  
)  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}  
learn>
```

Рисунок 1

```
> 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'
  }
}
```

Рисунок 2

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
}
> db.unicorns.find({name: "Pilot"})
< {
  _id: ObjectId('6836c159f8bf637f5e8f7c38'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
learn>
```

Рисунок 1-2

Практическое задание 3.3.7:

1. Изменить информацию о самке единорога Auroга: 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
}
```

Рисунок 1

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

Рисунок 2

Практическое задание 3.4.1:

1. 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"}}
```

2. Удалите документы с беспартийными мэрами.

3. Проверьте содержание коллекции.

4. Очистите коллекцию.

5. Просмотрите список доступных коллекций.

```
> db.towns.remove({"mayor.party": "I"})
< DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
< {
  acknowledged: true,
  deletedCount: 2
}
> db.towns.find()
< {
  _id: ObjectId('68373de3e192ff0624961060'),
  name: 'Punxsutawney',
  populatiuon: 6200,
  last_sensus: 2008-01-31T00:00:00.000Z,
  famous_for: [
    ''
  ],
  mayor: {
    name: 'Jim Wehrle'
  }
}
```

Рисунки 1 и 2

```
> db.towns.find().pretty()
< {
  _id: ObjectId('683765a22731fc2c90843f3a'),
  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('683765a22731fc2c90843f3b'),
  name: 'Portland',
  popujatiuon: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams',
    party: 'D'
  }
}
```

Рисунок 3


```

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

```

Рисунки 4 и 5

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: "Meadows"} ])
< {
  acknowledged: true,
  insertedIds: {
    '0': 'forest',
    '1': 'mountains',
    '2': 'meadows',
    '3': 'clouds'
  }
}

```

1 Рисунок

```
.. > db.unicorns.updateOne(  
    { name: "Horny" },  
    { $set: { habitat: { $ref: "habitats", $id: "forest" } } }  
  )  
  
db.unicorns.updateOne(  
  { name: "Aurora" },  
  { $set: { habitat: { $ref: "habitats", $id: "meadows" } } }  
)  
  
db.unicorns.updateOne(  
  { name: "Unicrom" },  
  { $set: { habitat: { $ref: "habitats", $id: "mountains" } } }  
)  
  
db.unicorns.updateOne(  
  { name: "Solnara" },  
  { $set: { habitat: { $ref: "habitats", $id: "forest" } } }  
)  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

2 Рисунок

```

> db.unicorns.find({ habitat: { $exists: true } }).pretty()
< {
  _id: ObjectId('6836c159f8bf637f5e8f7c2f'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 68,
  habitat: DBRef('habitats', 'forest')
}
{
  _id: ObjectId('6836c159f8bf637f5e8f7c30'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape',
    'sugar',
    'lemons'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43,
  habitat: DBRef('habitats', 'meadows')
}
{

```

3 Рисунок

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' ]
```

Рисунок 1

Ответ: 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>
```

Рисунок 1-3

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

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})}
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6837702e2731fc2c9085c5db')
  }
}
learn>
```

Рисунок 1

```
> db.numbers.find().sort({ _id: -1 }).limit(4)
< {
  _id: ObjectId('6837702e2731fc2c9085c5db'),
  value: 99999
}
{
  _id: ObjectId('6837702e2731fc2c9085c5da'),
  value: 99998
}
{
  _id: ObjectId('6837702e2731fc2c9085c5d9'),
  value: 99997
}
{
  _id: ObjectId('6837702e2731fc2c9085c5d8'),
  value: 99996
}
learn> |
```

Рисунок 2

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

Рисунок 4

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

Рисунок 5

```

]
> db.numbers.find({ value: { $gte: 99996 } }).sort({ value: 1 })
< {
  _id: ObjectId('6837702e2731fc2c9085c5d8'),
  value: 99996
}
{
  _id: ObjectId('6837702e2731fc2c9085c5d9'),
  value: 99997
}
{
  _id: ObjectId('6837702e2731fc2c9085c5da'),
  value: 99998
}
{
  _id: ObjectId('6837702e2731fc2c9085c5db'),
  value: 99999
}
learn> |
```

Рисунок 6

```
}  
> const indexedExplain = db.numbers.find({ value: { $gte: 99996 } }).sort({ value: 1 }).explain("executionStats")  
  print("Время выполнения с индексом (мс): " + indexedExplain.executionStats.executionTimeMillis)  
< Время выполнения с индексом (мс): 5  
learn>
```

7 Рисунок

Ответ: с индексом быстрее работает