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

Отчет

по лабораторной работе №6 «Работа с БД в СУБД MongoDB»

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Игнатьев А.А.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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

Оборудование: компьютерный класс.

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

Выполнение

Практическое задание 2.1.1:

1. *Создайте базу данных learn.*
2. *Заполните коллекцию единорогов unicorns:*

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

```
}
> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65a8fff051808db6900dbbc5')
    }
}
> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65a8fff351808db6900dbbc6')
    }
}
> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65a8fff651808db6900dbbc7')
    }
}
> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65a8fff951808db6900dbbc8')
    }
}
learn>
```

Используя второй способ, вставьте в коллекцию единорогов документ:

```
> 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('65a9003651808db6900dbbc9')
    }
  }
learn>
```

Проверьте содержимое коллекции с помощью метода find.

```
> db.unicorns.find()
< {
    _id: ObjectId('65a8ff9851808db6900dbbb3'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('65a8ff9851808db6900dbbb4'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('65a8ff9851808db6900dbbb5'),
    name: 'Unicrom',
    loves: [
      'energon',
      'redbull'
    ],
```

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.*

```
learn> db.unicorns.find({gender: "m"}).sort({name: "1"})
[
  {
    _id: ObjectId('65a90280c7dba939fddf3b7b'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65a9023fc7dba939fddf3b70'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65a9025ac7dba939fddf3b76'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('65a9026ac7dba939fddf3b79'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('65a90260c7dba939fddf3b77'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('65a9024dc7dba939fddf3b73'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('65a9024ac7dba939fddf3b72'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
learn>
```

```
learn> db.unicorns.find({gender: "f"}).sort({name: "1"}).limit(3)
[
  {
    _id: ObjectId('65a90246c7dba939fddf3b71'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65a90256c7dba939fddf3b75'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65a90263c7dba939fddf3b78'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn>
```

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

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
{
  _id: ObjectId('65a90246c7dba939fddf3b71'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
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'}, {gender: 0, loves: 0}).sort({name: 1})
[
  {
    _id: ObjectId('65a90280c7dba939fddf3b7b'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('65a9023fc7dba939fddf3b70'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('65a9025ac7dba939fddf3b76'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('65a9026ac7dba939fddf3b79'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('65a90260c7dba939fddf3b77'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('65a9024dc7dba939fddf3b73'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('65a9024ac7dba939fddf3b72'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
]
learn>
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

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: {$lt: 700, $gt: 500}}, {_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({weight: {$gt: 500}, loves:{$all:['grape', 'lemon']}}, {_id: 0})
[
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ `vampires`.

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId("658a0d6476fe46494ee1d29f"),
    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.

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

Практическое задание 3.1.1:

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

```
learn> db.towns.find()
[
  {
    _id: ObjectId('65a90bdbc7dba939fddf3b7c'),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('65a90be5c7dba939fddf3b7d'),
    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('65a90becc7dba939fddf3b7e'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn>
```

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"},{mayor:1, name:1, _id:0})
[
  {
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
```

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}},{mayor: 1, name: 1, _id:0})
[ { name: 'Punxsutawney ', mayor: { name: 'Jim Wehrle' } } ]
```

Практическое задание 3.1.2:

1. *Сформировать функцию для вывода списка самцов единорогов.*

```
learn> function printMaleUnicorms() {return db.unicorns.find({gender: 'm'})}
[Function: printMaleUnicorms]
learn> printMaleUnicorns()
ReferenceError: printMaleUnicorns is not defined
learn> printMaleUnicorns();
ReferenceError: printMaleUnicorns is not defined
learn> printMaleUnicorms();
[
  {
    _id: ObjectId('65a9023fc7dba939fddf3b70'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65a9024ac7dba939fddf3b72'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('65a9024dc7dba939fddf3b73'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
```

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

```
learn> var cursor = printMaleUnicorms().sort({name: 1}).limit(2);null;
null
learn> cursor.forEach(function(obj){print(obj)});
{
  _id: ObjectId('65a90280c7dba939fddf3b7b'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('65a9023fc7dba939fddf3b70'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}

learn>
```

Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг

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

Практическое задание 3.2.2:

Вывести список предпочтений.

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

Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

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

Практическое задание 3.3.1:

1. *Выполнить команду:*

```
learn> db.unicorns.save()
TypeError: db.unicorns.save is not a function
learn>
```

Практическое задание 3.3.2:

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

```
learn> db.unicorns.replaceOne({name: "Ayna"}, {name: "Ayna", loves: ["strawberry", "lemon"],
 weight: 800, gender: "f", vampires: 51})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
```

Проверить содержимое коллекции `unicorns`.

```
  _id: ObjectId("658a0d6376fe46494ee1d29a"),
  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`.

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

Практическое задание 3.3.4:

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

Проверить содержимое коллекции `unicorns`.

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

Практическое задание 3.3.5:

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

Проверить содержимое коллекции `towns`.

```
  {
    _id: ObjectId('65a90becc7dba939fddf3b7e'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
```

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

Практическое задание 3.3.6:

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

Проверить содержимое коллекции `unicorns`.

```
learn> db.unicorns.updateOne({name: 'Pilot'}, {$push: {loves: 'chocolate'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Pilot'})
[
  {
    _id: ObjectId("658a0d6476fe46494ee1d29e"),
    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.*

Проверить содержимое коллекции `unicorns`*.*

```
learn> db.unicorns.updateOne({name: 'Aurora'}, {$push: {loves: {$each:['sugar','lemon']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: 'Aurora'})
[
  {
    _id: ObjectId("658a0d6276fe46494ee1d296"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

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.*

```
learn> db.towns.find()
[
  {
    _id: ObjectId("658a2f8976fe46494ee1d2a1"),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId("658a2fb776fe46494ee1d2a2"),
    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("658a2fcf76fe46494ee1d2a3"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  },
  {
    _id: ObjectId("658b5b0a76fe46494ee1d2a6"),
    name: 'Punxsutawney ',
    popujatiuon: 6200,
    last_sensus: ISODate("2008-01-31T00:00:00.000Z"),
    famous_for: [ 'phil the groundhog' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId("658b5b4376fe46494ee1d2a7"),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate("2009-07-31T00:00:00.000Z"),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
learn>
```

```
learn> db.towns.deleteMany({'mayor.party':{$exists: false}})
{ acknowledged: true, deletedCount: 3 }
```

```
learn> db.towns.find()
[
  {
    _id: ObjectId("658a2fb776fe46494ee1d2a2"),
    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("658b5b4376fe46494ee1d2a7"),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate("2009-07-31T00:00:00.000Z"),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId("658b5b6776fe46494ee1d2a8"),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate("2009-07-20T00:00:00.000Z"),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
```

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

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 едиорогов.*

```
learn>  db.places.insertMany([{"_id": "forest", "name": "Forest", "description": "coming so
 {"_id": "dreams", "name": "Dreams", "description": "coming later"}])
{ acknowledged: true, insertedIds: { '0': 'forest', '1': 'dreams' } }
```

```
var forestId = db.places.findOne({_id: "forest"})._id

var dreamsId = db.places.findOne({_id: "dreams"})._id
```

```
learn> db.unicorns.updateMany({name: "Ayna"},{$set: {places: {$ref: "places", $id: forestId}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.updateMany({name: "Aurora"},{$set: {places: {$ref: "places", $id: dreamsId}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
learn> db.unicorns.find({name: 'Ayna'})
[
  {
    _id: ObjectId("658a0d6376fe46494ee1d29a"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51,
    places: DBRef("places", 'forest')
  }
]
learn>  db.unicorns.find({name: 'Aurora'})
[
  {
    _id: ObjectId("658a0d6276fe46494ee1d296"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    places: DBRef("places", 'dreams')
  }
]
```

Практическое задание 4.2.1:

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

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

Практическое задание 4.3.1:

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

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

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

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

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* `numbres`*.*

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?*

```
learn> db.createCollection("numbers")
{ ok: 1 }
learn> for(i=0; i < 100000; i++){db.numbers.insert({value:i})}


{
  acknowledged: true,
  insertedIds: { '0': ObjectId("658b815476fe46494ee4dfe8") }
}
```

```
earn> db.numbers.find().count()
00000
```

```
learn>  db.numbers.find().sort({value: -1}).limit(4).explain("executionStats")
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 122,
  totalKeysExamined: 0,
  totalDocsExamined: 100000,
```

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

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

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 6,
  totalKeysExamined: 4,
  totalDocsExamined: 4,
```

Запрос после добавления индекса стал быстрее

Вывод

В ходе лабораторной работы я получил практические навыки работы с Mongo BD, изучил инструментарий CRUD операций, и основные команды работы с БД