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»

Обучающийся Скобликов Кирилл Александрович
Факультет прикладной информатики
Группа K3239
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

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

Практическое задание 2.1.1:

1. *Создайте базу данных learn.*
2. *Заполните коллекцию единорогов unicorns представленными данными*
3. *Используя второй способ, вставить в коллекцию единорогов документ*
4. *Проверить содержимое коллекции с помощью метода find.*

Выполнение:

```
test> use learn
switched to db learn
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: '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'});
...
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68236263392ef11d096c4bda') }
}
learn> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6823627c392ef11d096c4bdb') }
}
```

```
learn> doc = {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(doc)
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6823631d392ef11d096c4bdc') }
}
learn> db.unicorns.find()
[
  {
    _id: ObjectId('68236263392ef11d096c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
```

```
{
  _id: ObjectId('68236263392ef11d096c4bd2'),
  name: 'Unicrom',
  loves: [ 'energon', 'redbull' ],
  weight: 984,
  gender: 'm',
  vampires: 182
},
{
  _id: ObjectId('68236263392ef11d096c4bd3'),
  name: 'Roooooodles',
  loves: [ 'apple' ],
  weight: 575,
  gender: 'm',
  vampires: 99
},
{
  _id: ObjectId('68236263392ef11d096c4bd4'),
  name: 'Solnara',
  loves: [ 'apple', 'carrot', 'chocolate' ],
  weight: 550,
  gender: 'f',
  vampires: 80
},
{
  _id: ObjectId('68236263392ef11d096c4bd5'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 733,
  gender: 'f',
  vampires: 40
},
```

```
{
  _id: ObjectId('68236263392ef11d096c4bd5'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 733,
  gender: 'f',
  vampires: 40
},
{
  _id: ObjectId('68236263392ef11d096c4bd6'),
  name: 'Kenny',
  loves: [ 'grape', 'lemon' ],
  weight: 690,
  gender: 'm',
  vampires: 39
},
{
  _id: ObjectId('68236263392ef11d096c4bd7'),
  name: 'Raleigh',
  loves: [ 'apple', 'sugar' ],
  weight: 421,
  gender: 'm',
  vampires: 2
},
{
  _id: ObjectId('68236263392ef11d096c4bd8'),
  name: 'Leia',
  loves: [ 'apple', 'watermelon' ],
  weight: 601,
  gender: 'f',
  vampires: 33
},
{
  _id: ObjectId('68236263392ef11d096c4bd9'),
  name: 'Pilot',
  loves: [ 'apple', 'watermelon' ],
  weight: 650,
  gender: 'm',
  vampires: 54
},
```

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.*

Выполнение:

```
learn> db.unicorns.find({gender: "m"}).sort({name: 1})
[
  {
    _id: ObjectId('6823631d392ef11d096c4bdc'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6823627c392ef11d096c4bdb'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
```

```
[
  {
    _id: ObjectId('68236263392ef11d096c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
]
```

```
learn> db.unicorns.find({gender: "f"}).sort({name: 1}).limit(3)
[
  {
    _id: ObjectId('68236263392ef11d096c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn> db.unicorns.findOne({gender: "f", loves: "carrot"})
{
  _id: ObjectId('68236263392ef11d096c4bd1'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Практическое задание 2.2.2:

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

Выполнение:

```
learn> db.unicorns.find({gender: "m"}, {loves: 0, gender: 0}).sort({name: 1})
[
  {
    _id: ObjectId('6823631d392ef11d096c4bdc'),
    name: 'Dunx',
    weight: 704,
    vampires: 165
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd0'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('6823627c392ef11d096c4bdb'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd6'),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd9'),
    name: 'Pilot',
    weight: 650,
    vampires: 54
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd7'),
    name: 'Raleigh',
    weight: 421,
    vampires: 2
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd3'),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd2'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

Выполнение:

```
learn> db.unicorns.find().sort({ $natural: -1 })
[
  {
    _id: ObjectId('6823631d392ef11d096c4bdc'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6823627c392ef11d096c4bdb'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68236263392ef11d096c4bda'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd8'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
```

```
{
  _id: ObjectId('68236263392ef11d096c4bd7'),
  name: 'Raleigh',
  loves: [ 'apple', 'sugar' ],
  weight: 421,
  gender: 'm',
  vampires: 2
},
{
  _id: ObjectId('68236263392ef11d096c4bd6'),
  name: 'Kenny',
  loves: [ 'grape', 'lemon' ],
  weight: 690,
  gender: 'm',
  vampires: 39
},
{
  _id: ObjectId('68236263392ef11d096c4bd5'),
  name: 'Ayna',
  loves: [ 'strawberry', 'lemon' ],
  weight: 733,
  gender: 'f',
  vampires: 40
},
{
  _id: ObjectId('68236263392ef11d096c4bd4'),
  name: 'Solnara',
  loves: [ 'apple', 'carrot', 'chocolate' ],
  weight: 550,
  gender: 'f',
  vampires: 80
},
{
  _id: ObjectId('68236263392ef11d096c4bd3'),
  name: 'Roooooodles',
  loves: [ 'apple' ],
  weight: 575,
  gender: 'm',
  vampires: 99
},
{
  _id: ObjectId('68236263392ef11d096c4bd2'),
  name: 'Unicrom',
  loves: [ 'energon', 'redbull' ],
  weight: 984,
  gender: 'm',
  vampires: 182
},
{
  _id: ObjectId('68236263392ef11d096c4bd1'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
},
{
  _id: ObjectId('68236263392ef11d096c4bd0'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

Практическое задание 2.2.4:

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

Выполнение:

```
learn> db.unicorns.find({}, { loves: {$slice: 1}, _id: 0})
[
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Aurora',
    loves: [ 'carrot' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    name: 'Unicrom',
    loves: [ 'energon' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    name: 'Solnara',
    loves: [ 'apple' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
```

```
{
  name: 'Ayna',
  loves: [ 'strawberry' ],
  weight: 733,
  gender: 'f',
  vampires: 40
},
{
  name: 'Kenny',
  loves: [ 'grape' ],
  weight: 690,
  gender: 'm',
  vampires: 39
},
{
  name: 'Raleigh',
  loves: [ 'apple' ],
  weight: 421,
  gender: 'm',
  vampires: 2
},
{
  name: 'Leia',
  loves: [ 'apple' ],
  weight: 601,
  gender: 'f',
  vampires: 33
},
{
  name: 'Pilot',
  loves: [ 'apple' ],
  weight: 650,
  gender: 'm',
  vampires: 54
},
{ name: 'Nimue', loves: [ 'grape' ], weight: 540, gender: 'f' },
 {
   name: 'Horny',
   loves: [ 'carrot' ],
   weight: 600,
   gender: 'm',
   vampires: 63
 },
 {
   name: 'Dunx',
   loves: [ 'grape' ],
   weight: 704,
   gender: 'm',
   vampires: 165
 }
```

Практическое задание 2.3.1:

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

Выполнение:

```
learn> db.unicorns.find({gender: "f", weight: { $gte: 500, $lt: 700}}, {_id: 0}).sort({name: 1})
[
  {
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
```

Практическое задание 2.3.2:

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

Выполнение:

```
learn> db.unicorns.find({weight: {$gte: 500}, loves: {$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.

Выполнение:

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('68236263392ef11d096c4bda'),
    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.

Выполнение:

```
learn> db.unicorns.find({gender: "m"}, {name: 1, _id: 0, loves: {$slice: 1}}).sort({name: 1})
[
  { name: 'Dunx', loves: [ 'grape' ] },
  { name: 'Horny', loves: [ 'carrot' ] },
  { 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. *Создать коллекцию towns, включающую представленные документы*
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 мэрею*

Выполнение:

```
learn> db.towns.insert({name: "Punxsutawney ",
... populatiuon: 6200,
... last_sensus: ISODate("2008-01-31"),
... famous_for: [""],
... mayor: {
...     name: "Jim Wehrle"
... }}
... )
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68237e87392ef11d096c4bdd') }
}
learn> db.towns.insert({name: "New York",
... populatiuon: 22200000,
... last_sensus: ISODate("2009-07-31"),
... famous_for: ["status of liberty", "food"],
... mayor: {
...     name: "Michael Bloomberg",
... party: "I"}}
... )
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68237ea5392ef11d096c4bde') }
}
learn> db.towns.insert({name: "Portland",
... populatiuon: 528000,
... last_sensus: ISODate("2009-07-20"),
... famous_for: ["beer", "food"],
... mayor: {
...     name: "Sam Adams",
... party: "D"}}
... )
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68237ec5392ef11d096c4bdf') }
}
learn> db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 1})
[
  {
    _id: ObjectId('68237ea5392ef11d096c4bde'),
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
learn> db.towns.find({"mayor.party": {$exists: 0}}, {name: 1, mayor: 1})
[
  {
    _id: ObjectId('68237e87392ef11d096c4bdd'),
    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.*

Выполнение:

```
learn> fn = function() {return this.gender=="m";}
[Function: fn]
learn> db.unicorns.find({ $where: fn})
[
  {
    _id: ObjectId('68236263392ef11d096c4bd0'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd2'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd3'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd6'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
```

```
  {
    _id: ObjectId('68236263392ef11d096c4bd7'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd9'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6823627c392ef11d096c4bdb'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('6823631d392ef11d096c4bdc'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
]
learn> var cursor = db.unicorns.find(); null
null
learn> cursor.limit(2); null
null
learn> cursor.sort({name: 1})
[
  {
    _id: ObjectId('68236263392ef11d096c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('68236263392ef11d096c4bd5'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
]
```

Практическое задание 3.2.1:
Вывести количество самок единорогов весом от полутонны до 600 кг.
Выполнение:

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

Практическое задание 3.2.2:
Вывести список предпочтений.
Выполнение:

```
learn> 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.
Выполнение:

```
learn> db.unicorns.aggregate({"$group": {_id: "$gender", count: {$sum: 1}}})
[ { _id: 'f', count: 5 }, { _id: 'm', count: 8 } ]
```

Практическое задание 3.3.1:

1. *Выполнить команду:*

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

2. *Проверить содержимое коллекции* `unicorns`.

Выполнение:

При попытке вставить команду, вылезла ошибка:

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

Практическое задание 3.3.2:

1. *Для самки единорога* `Ayna` *внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.*
2. *Проверить содержимое коллекции* `unicorns`.

Выполнение:

```
learn> db.unicorns.update({name: "Ayna"}, {$set: {weight: 800, vampires: 51}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: "Ayna"})
[
  {
    _id: ObjectId('68236263392ef11d096c4bd5'),
    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.*
2. *Проверить содержимое коллекции* `unicorns`.

Выполнение:

```
learn> db.unicorns.update({name: "Raleigh"}, {$set: {loves: "AdrenalinRush"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: "Raleigh"})
[
  {
    _id: ObjectId('68236263392ef11d096c4bd7'),
    name: 'Raleigh',
    loves: 'AdrenalinRush',
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

Практическое задание 3.3.4:

1. *Всем самцам единорогов увеличить количество убитых вапмиров на 5.*
2. *Проверить содержимое коллекции* `unicorns`.

Выполнение:

```
learn> db.unicorns.update({gender: "m"}, {$inc: {vampires: 5}}, {multi: true})
{
  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.*
2. *Проверить содержимое коллекции* `towns`.

Выполнение:

```
learn> db.towns.update({name: "Portland"}, {$unset: {"mayor.party": 1}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.towns.find()
[
  {
    _id: ObjectId('68237e87392ef11d096c4bdd'),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: ISODate('2008-01-31T00:00:00.000Z'),
    famous_for: [ '' ],
    mayor: { name: 'Jim Wehrle' }
  },
  {
    _id: ObjectId('68237ea5392ef11d096c4bde'),
    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('68237ec5392ef11d096c4bdf'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams' }
  }
]
```

Практическое задание 3.3.6:

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

Выполнение:

```
learn> 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('68236263392ef11d096c4bd9'),
    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`.

Выполнение:

```
learn> db.unicorns.update({name: "Aurora"}, {$addToSet: {loves: {$each: ["sugar", "lemon"]}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: "Aurora"})
[
  {
    _id: ObjectId('68236263392ef11d096c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 3.4.1:

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

Выполнение:

```
learn> db.towns.remove({"mayor.party": {$exists: false}})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 2 }
learn> db.towns.find()
[
  {
    _id: ObjectId('68237ea5392ef11d096c4bde'),
    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' }
  }
]
learn> db.towns.remove({})
{ acknowledged: true, deletedCount: 1 }
learn> show collections
towns
unicorns
```

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.*

Выполнение:

```
learn> db.uhone.insert({_id: "Bol", name: "Boloto", opisanie: "Gluboko i vonyaet"})
{ acknowledged: true, insertedIds: { '0': 'Bol' } }
learn> db.uhone.insert({_id: "Pus", name: "PustbInya", opisanie: "Zarko ochenb"})
{ acknowledged: true, insertedIds: { '0': 'Pus' } }
learn> db.unicorns.update({name: "Aurora"}, {$set: {home:{$ref:"uhone", $id: "Bol"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.update({name: "Pilot"}, {$set: {home:{$ref:"uhone", $id: "Pus"}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name: "Aurora"})
[
  {
    _id: ObjectId('68236263392ef11d096c4bd1'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43,
    home: DBRef('uhone', 'Bol')
  }
]
```

Практическое задание 4.2.1:

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

Выполнение:

```
learn> db.unicorns.ensureIndex({"name": 1}, {"unique": true})
MongoServerError[DuplicateKey]: Index build failed: b68b1ad8-dc91-4517-9d8e-85575fa940e4: Collection learn.unicorns
 :: caused by :: E11000 duplicate key error collection: learn.unicorns index: name_1 dup key: { name: "Horny" }
```

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.*

Выполнение:

```
learn> db.unicorns.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn> db.unicorns.dropIndexes("_id")
MongoServerError[IndexNotFound]: index not found with name [_id]
```

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?*

Выполнение:

```
learn> for(i=20; i < 100000; i++) {db.numbers.insert({value: i})}
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6823a52d392ef11d096f085e') }
}
learn> db.numbers.find({value: {$gte: 99996}})
[
  { _id: ObjectId('6823a52d392ef11d096f085b'), value: 99996 },
  { _id: ObjectId('6823a52d392ef11d096f085c'), value: 99997 },
  { _id: ObjectId('6823a52d392ef11d096f085d'), value: 99998 },
  { _id: ObjectId('6823a52d392ef11d096f085e'), value: 99999 }
]
learn> db.numbers.explain("executionStats").find({value: {$gte: 99996}})
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: { value: { '$gte': 99996 } },
    indexFilterSet: false,
    queryHash: 'AA258D80',
    planCacheShapeHash: 'AA258D80',
    planCacheKey: 'C5B6FEE0',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { value: { '$gte': 99996 } },
      direction: 'forward'
    },
    rejectedPlans: []
```

```
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 124,
    totalKeysExamined: 0,
    totalDocsExamined: 179327,
    executionStages: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { value: { '$gte': 99996 } },
      nReturned: 4,
      executionTimeMillisEstimate: 119,
      works: 179328,
      advanced: 4,
      needTime: 179323,
      needYield: 0,
      saveState: 7,
      restoreState: 7,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 179327
    }
  },
  queryShapeHash: 'AF8E09276B8C4FF9FBBAA4ACB7DA4D92C8E99DADA95E5977ED7C6FBA55D93BBA',
  command: {
    find: 'numbers',
    filter: { value: { '$gte': 99996 } },
    '$db': 'learn'
  },
  serverInfo: {
    host: 'FLTP-5i5-16512',
    port: 27017,
    version: '8.0.9',
    gitVersion: 'f882ef816d531ecfbb593843e4c554fda90ca416'
  },
```

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

```
learn> db.numbers.explain("executionStats").find({value: {$gte: 99996}})
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: { value: { '$gte': 99996 } },
    indexFilterSet: false,
    queryHash: 'AA258D80',
    planCacheShapeHash: 'AA258D80',
    planCacheKey: 'C38141D5',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { value: 1 },
        indexName: 'value_1',
        isMultiKey: false,
        multiKeyPaths: { value: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { value: [ '[99996, inf.0]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 13,
    totalKeysExamined: 4,
    totalDocsExamined: 4,
```

Более эффективен запрос 2

Вывод: в ходе работы удалось 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.