

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: Цыпандин А.П.

Факультет: ИКТ

Группа: K3239

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

ЦЕЛЬ РАБОТЫ.....	3
ПРАКТИЧЕСКОЕ ЗАДАНИЕ	3
ВАРИАНТ 14. БД «СЛУЖБА ЗАКАЗА ТАКСИ».....	3
ВЫПОЛНЕНИЕ	3
Задания.....	5
ВЫВОД	4

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:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов.
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

Выполнение работы

Практическое задание лабораторной работы 6.1:

```
[test> use learn
switched to db learn
[learn> db.createCollection('unicorns');
{ ok: 1 }
[learn> db.unicorns.insertOne({name: 'Aurora', gender: 'f', weight: 450});
{
  acknowledged: true,
  insertedId: ObjectId('65ba2856cabe648a8ac079b6')
}
[learn> show collections
unicorns
[learn> db.unicorns.renameCollection('unicorn');
{ ok: 1 }
[learn> db.unicorn.stats();
{
  ok: 1,
```

```
[learn> db.unicorn.drop();
true
[learn> db.dropDatabase();
{ ok: 1, dropped: 'learn' }
```

Практическое задание 2.1.1:

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

```
learn> db.unicorns.insert({name: 'Horny', loves: ['carrot', 'papaya'], weight: 600, gender: 'm', vampires: 63});
elon', weight: 650, gender: 'm', vampires: 54});
db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079b7') }
}
learn> db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079b8') }
}
learn> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079b9') }
}
learn> db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079ba') }
}
learn> db.unicorns.insert({name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079bb') }
}
learn> db.unicorns.insert({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079bc') }
}
learn> db.unicorns.insert({name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079bd') }
}
learn> db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079be') }
}
learn> db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079bf') }
}
learn> db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079c0') }
}
learn> db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba486ccabe648a8ac079c1') }
}
}
```

3. *Используя второй способ, вставьте в коллекцию единорогов документ*
4. *Проверьте содержимое коллекции с помощью метода find.*

```
learn> document = ({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(document);
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('65ba4898cabe648a8ac079c2') }
}
learn> db.unicorns.find();
[
  {
    _id: ObjectId('65ba486ccabe648a8ac079b7'),
```

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.

```
learn> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3);
[
  {
    _id: ObjectId('65ba486ccabe648a8ac079b8'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('65ba486ccabe648a8ac079bc'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('65ba486ccabe648a8ac079bf'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
]
learn> db.unicorns.find({gender: 'm'}).sort({name: 1}).limit(3);
[
  {
    _id: ObjectId('65ba4898cabe648a8ac079c2'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65ba486ccabe648a8ac079b7'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('65ba486ccabe648a8ac079bd'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
learn> 
```

```
learn> db.unicorns.findOne({gender: 'f', loves: 'carrot'});
{
  _id: ObjectId('65ba486ccabe648a8ac079b8'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1);
[
  {
    _id: ObjectId('65ba486ccabe648a8ac079b8'),
    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 предпочтениях и поле.

```
[learn> db.unicorns.find({gender: 'f'}, {loves: 0, gender: 0});
[
  {
    _id: ObjectId('65ba486ccabe648a8ac079b8'),
    name: 'Aurora',
    weight: 450,
    vampires: 43
  },
  {
    _id: ObjectId('65ba486ccabe648a8ac079bb'),
    name: 'Solnara',
    weight: 550,
    vampires: 80
  },
  {
    _id: ObjectId('65ba486ccabe648a8ac079bc'),
    name: 'Ayna',
    weight: 733,
    vampires: 40
  },
  {
    _id: ObjectId('65ba486ccabe648a8ac079bf'),
    name: 'Leia',
    weight: 601,
    vampires: 33
  },
  {
    _id: ObjectId('65ba486ccabe648a8ac079c1'),
    name: 'Nimue',
    weight: 540
  }
]
```

Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.

```
[learn> db.unicorns.find().sort({ $natural: -1 });
[
  {
    _id: ObjectId('65ba4898cabe648a8ac079c2'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('65ba486ccabe648a8ac079c1'),
    name: 'Nimue',
    loves: [ 'apple', 'orange' ],
    weight: 540,
    gender: 'f',
    vampires: 40
  }
]
```

Практическое задание 2.2.4:

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

```
[learn> db.unicorns.find({}, {_id: 0, loves: {$slice: 1}});
[
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Aurora',
    loves: [ 'carrot' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 2.3.1:

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

```
learn> db.unicorns.find({weight: {$gt: 500, $lt: 700}}, {_id: 0});
[
  {
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Rooooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
learn> 
```

Практическое задание 2.3.2:

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

```
learn> db.unicorns.find({"gender": "m", "weight": {"$gte": 500}, "loves": {"$all": ["grape", "lemon"]}}, {_id: false});
[
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 43
  }
]
```

Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires.

```
learn> db.unicorns.find({vampires: {$exists: false}});
[
  {
    _id: ObjectId('65ba486ccabe648a8ac079c1'),
    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({}, {_id: 0, loves: {$slice: 1}, weight: 0, gender: 0, vampires: 0}).sort({name: 1});
[
  { name: 'Aurora', loves: [ 'carrot' ] },
  { name: 'Ayna', loves: [ 'strawberry' ] },
  { name: 'Dunx', loves: [ 'grape' ] },
  { name: 'Horny', loves: [ 'carrot' ] },
  { name: 'Kenny', loves: [ 'grape' ] },
  { name: 'Leia', loves: [ 'apple' ] },
  { name: 'Nimue', loves: [ 'grape' ] },
  { name: 'Pilot', loves: [ 'apple' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Roooooodles', loves: [ 'apple' ] },
  { name: 'Solnara', loves: [ 'apple' ] },
  { name: 'Unicrom', loves: [ 'energon' ] }
]
```

Практическое задание 3.1.1:

1) Создайте коллекцию towns, включающую следующие документы:

```
learn> db.createCollection('towns');
{ ok: 1 }
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('65bb59dbcabe648a8ac079c3') } }
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('65bb59eecake648a8ac079c4') } }
learn> db.towns.insert({name: "Portland",
```

2)

```
learn> db.towns.find({'mayor.party': 'I'});
[
  {
    _id: ObjectId('65bb59eecake648a8ac079c4'),
    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> █
```

3)

```
learn> db.towns.find({'mayor.party': {$exists: false}}, {_id: 0, population: 0, last_sensus: 0, famous_for: 0});
[
  {
    name: 'Punxsutawney ',
    populatiuon: 6200,
    mayor: { name: 'Jim Wehrle' }
  }
]
learn> █
```

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.


```
[learn> const find_all_male_unicorns = () => db.unicorns.find({"gender": "m"})

[learn> var cursor = find_all_male_unicorns().sort({name: 1}).limit(2);
```

```
[learn> cursor.forEach((unicorn) => print(unicorn));
{
  _id: ObjectId('65ccb1fece77381d384816bf'),
  name: 'Barny',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm',
  vampires: 4,
  area: DBRef('areas', 'mtn')
}
{
  _id: ObjectId('65ba4898cabe648a8ac079c2'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 169,
  area: DBRef('areas', 'lake')
}
```

Практическое задание 3.2.1:

```
[learn> db.unicorns.find({weight: {$lt: 600}}).count();
5
[learn> db.unicorns.find({weight: {$lte: 600}}).count();
6
```

Практическое задание 3.2.2:

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

Практическое задание 3.2.3:

```
learn> db.unicorns.find({gender: 'm'}).count();
7
learn> db.unicorns.find({gender: 'f'}).count();
5
```

Практическое задание 3.3.1:

NOTE:

Starting in MongoDB 4.2, the `db.collection.save()` method is deprecated. Use `db.collection.insertOne()` or `db.collection.replaceOne()` instead.

```
learn> db.unicorns.insertOne({name: 'Barney', loves: ['grape'], weight: 340, gender: 'm'});
{
  acknowledged: true,
  insertedId: ObjectId('65bbb0d6cabe648a8ac079c6')
}
learn> █
```

Практическое задание 3.3.2:

! IMPORTANT

Deprecated mongosh Method

This method is deprecated in [mongosh](#). For alternative methods, see [Compatibility Changes with Legacy mongo Shell](#).

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

Практическое задание 3.3.3:

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

Практическое задание 3.3.4:

```
learn> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 4}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
learn> █
```

Практическое задание 3.3.5:

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

Практическое задание 3.3.6:

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

Практическое задание 3.3.7:

```
learn> db.unicorns.find({name: 'Aurora'});
[
  {
    _id: ObjectId('65ba486ccabe648a8ac079b8'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
learn> 
```

Практическое задание 3.4.1:

```
learn> db.towns.insertOne({name: "New York",
... popujatiuon: 22200000,
... last_sensus: ISODate("2009-07-31"),
... famous_for: ["status of liberty", "food"],
... mayor: {
...   name: "Michael Bloomberg",
...   party: "I"}}
{ acknowledged: true,
  insertedId: ObjectId('65bbcea5cabe648a8ac079cb') }
learn> db.towns.insertOne({name: "Portland",
... popujatiuon: 528000,
... last_sensus: ISODate("2009-07-20"),
... famous_for: ["beer", "food"],
... mayor: {
...   name: "Sam Adams",
...   party: "D"}}
{ acknowledged: true,
  insertedId: ObjectId('65bbceb0cabe648a8ac079cc') }
learn> 
```

```
learn> db.towns.deleteMany({'mayor.party': { $exists: false }});
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find();
[
  {
    _id: ObjectId('65bbcea5cabe648a8ac079cb'),
    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('65bbceb0cabe648a8ac079cc'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
  }
]
learn> 
```

```
learn> db.towns.drop();
true
learn> db.getCollectionNames();
[ '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.

```

learn> db.unicorns.update({_id: ObjectId('65ccb1fece77381d384816bf')}, {$set: {area:{$ref:'areas', $id: 'mtn'}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.update({_id: ObjectId("65ba4898cabe648a8ac079c2")}, {$set: {area:{$ref:'areas', $id: 'lake'}}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

```

```

{
  _id: ObjectId('65ba4898cabe648a8ac079c2'),
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 169,
  area: DBRef('areas', 'lake')
},
{
  _id: ObjectId('65ccb1fece77381d384816bf'),
  name: 'Barny',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm',
  vampires: 4,
  area: DBRef('areas', 'mtn')
}
]

```

Практическое задание 4.2.1:

1. Получите информацию о всех индексах коллекции `unicorns`.

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

```
[learn> db.unicorns.dropIndex("name_1");  
{ nIndexesWas: 2, ok: 1 }  
[learn> db.unicorns.dropIndex("_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.

```
[learn> db.numbers.find().sort({_id: -1}).limit(4);  
[  
  { _id: ObjectId('65ccc434ce77381d38499d5f'), value: 99999 },  
  { _id: ObjectId('65ccc434ce77381d38499d5e'), value: 99998 },  
  { _id: ObjectId('65ccc434ce77381d38499d5d'), value: 99997 },  
  { _id: ObjectId('65ccc434ce77381d38499d5c'), value: 99996 }  
]
```

3. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра `executionTimeMillis`)

```

learn> db.numbers.find().sort({_id: -1}).limit(4).explain("executionStats");
{
  explainVersion: '2',
  queryPlanner: {
    namespaces: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: 'FFF34A83',
    planCacheKey: 'DD8B8806',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      queryPlan: {
        stage: 'LIMIT',
        planNodeId: 3,
        limitAmount: 4,
        inputStage: {
          stage: 'FETCH',
          planNodeId: 2,
          inputStage: {
            stage: 'IXSCAN',
            planNodeId: 1,
            keyPattern: { _id: 1 },
            indexName: '_id_',
            isMultiKey: false,
            isUnique: true,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'backward',
            indexBounds: { _id: [ '[MaxKey, MinKey]' ] }
          }
        }
      },
      slotBasedPlan: {
        slots: '$$RESULT=s9 env: { s3 = 1707918494530 (NOW), s1 = TimeZoneDatabase(Brazil/Acre...Antarctica/DumontDUrville) (timeZoneDB), s2 = Nothing (SEARCH_META), s8 = { "_id" : 1 } }',
        stages: '[3] limit 4 \n' +
          '[2] nlj inner [1] [s4, s5, s6, s7, s8] \n' +
          '  left \n' +
          '    [1] ixseek KS(F0FE04) KS(0A0104) s7 s4 s5 s6 lowPriority [1] @"3a16aaa7-4366-46ae-bd6c-6c4445a02c26" @"_id_" false \n' +
          '    right \n' +
          '      [2] limit 1 \n' +
          '      [2] seek s4 s9 s10 s5 s6 s7 s8 [1] @"3a16aaa7-4366-46ae-bd6c-6c4445a02c26" true false \n'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 8,
  }
}

```

4. Создайте индекс для ключа *value*.
5. Получите информацию о всех индексах коллекции *numbers*.
6. Выполните запрос 2.

```

[learn> db.numbers.ensureIndex({"values": 1});
[ 'values_1' ]
[learn> db.numbers.getIndexes();
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { values: 1 }, name: 'values_1' }
]
[learn> db.numbers.find().sort({_id: -1}).limit(4);
[
  { _id: ObjectId('65ccc434ce77381d38499d5f'), value: 99999 },
  { _id: ObjectId('65ccc434ce77381d38499d5e'), value: 99998 },
  { _id: ObjectId('65ccc434ce77381d38499d5d'), value: 99997 },
  { _id: ObjectId('65ccc434ce77381d38499d5c'), value: 99996 }
]

```

7. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?

```
learn> db.numbers.find().sort({_id: -1}).limit(4).explain("executionStats");
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'learn.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: 'FFF34A83',
    planCacheKey: 'FD893160',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      queryPlan: {
        stage: 'LIMIT',
        planNodeId: 3,
        limitAmount: 4,
        inputStage: {
          stage: 'FETCH',
          planNodeId: 2,
          inputStage: {
            stage: 'IXSCAN',
            planNodeId: 1,
            keyPattern: { _id: 1 },
            indexName: '_id',
            isMultiKey: false,
            isUnique: true,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'backward',
            indexBounds: { _id: [ '[MaxKey, MinKey]' ] }
          }
        }
      },
      slotBasedPlan: {
        slots: '3$RESULT=s9 env: { s2 = Nothing (SEARCH_META), s8 = { "_id" : 1 }, s3 = 1707918674411 (NOW), s1 = TimeZoneDatabase(Brazil/Acre...Antarctica/DumontD'Urville) (timeZoneDB) }',
        stages: '[3] limit 4 \n' +
          '[2] nlj inner [] [s4, s5, s6, s7, s8] \n' +
          '  left \n' +
          '    [1] ixseek KS(F0FE94) KS(0A0104) s7 s4 s5 s6 lowPriority [] @*3a16aaa7-4366-46ae-bd6c-6c4445a82c26" @"_id_" false \n' +
          '    right \n' +
          '      [2] limit 1 \n' +
          '      [2] seek s4 s9 s10 s5 s6 s7 s8 [] @*3a16aaa7-4366-46ae-bd6c-6c4445a82c26" true false \n'
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 4,
      executionTimeMillis: 1,
    }
  }
}
```

8. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

В запросе с индексом значение ExecutionTimeMillis = 1 мс, а в запросе без индекса значение в восемь раз выше ExecutionTimeMillis = 8 мс. Очевидно, что запрос с индексом работает гораздо быстрее.

Вывод:

В этой лабораторной работе были проделаны практические работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.