Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Кафедра вычислительных систем

ОТЧЕТ
по лабораторной работе № 6 на тему:
«Работа с БД в СУБД MongoDB**»**

Выполнил: студент группы К3240
ФИО: Васильев Артур Дмитриевич

Проверил: преподаватель М.М. Говорова

Санкт-Петербург
2025

Цель работы

Овладеть практическими навыками работы с CRUD-операциями, вложенными объектами, агрегацией, изменением данных, ссылками и индексами в MongoDB.

Практическое задание 2.1.1

Решение:

```
> 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('6857fad8088eb32201c0ddde')
    }
  }
> 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('6857fc97088eb32201c0dddf')
    }
  }
> db.unicorns.find()
< {
    _id: ObjectId('6857fad8088eb32201c0ddd4'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
```

Практическое задание 2.2.1

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

Решение:

```
> db.unicorns.find({gender: 'm'}).sort({name: 1})
< {
    _id: ObjectId('6857fc97088eb32201c0dddf'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddd4'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddda'),
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
  {
    _id: ObjectId('6857fad8088eb32201c0dddd'),
    name: 'Pilot',
```

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
< {
    _id: ObjectId('6857fad8088eb32201c0ddd5'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddd9'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
    _id: ObjectId('6857fad8088eb32201c0dddc'),
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
```

```
> db.unicorns.find({gender: 'f', loves: 'carrot'})
< {
    _id: ObjectId('6857fad8088eb32201c0ddd5'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddd8'),
    name: 'Solnara',
    loves: [
      'apple',
      'carrot',
      'chocolate'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddde'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
< {
    _id: ObjectId('6857fad8088eb32201c0ddd5'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
< {
    _id: ObjectId('6857fad8088eb32201c0ddd5'),
    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 предпочтениях и поле.

Решение:

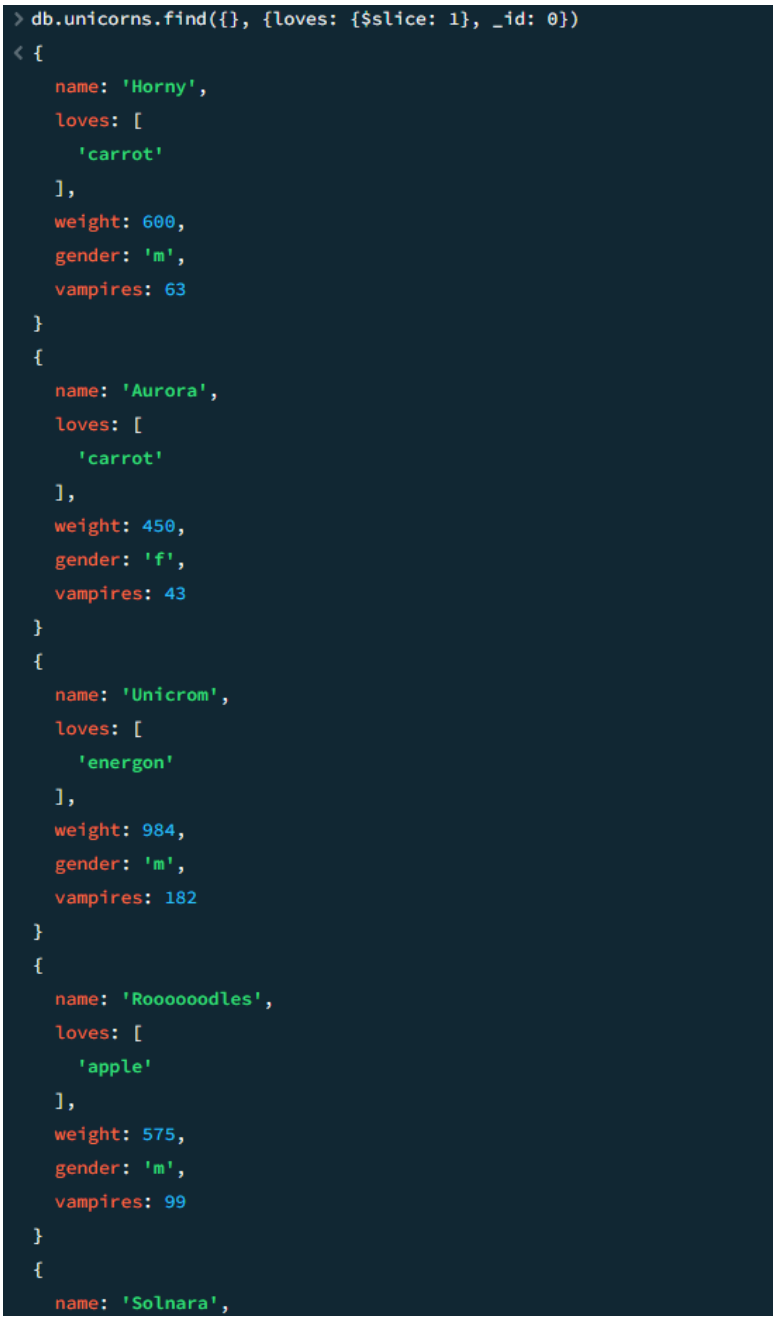
Практическое задание 2.2.3
Вывести список единорогов в обратном порядке добавления.

Решение:

```
> db.unicorns.find().sort({$natural: -1})
< {
    _id: ObjectId('6857fc97088eb32201c0dddf'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddde'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
  {
    _id: ObjectId('6857fad8088eb32201c0dddd'),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
  {
    _id: ObjectId('6857fad8088eb32201c0dddc'),
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
```

Практическое задание 2.1.4
Вывести список единорогов с названием первого любимого предпочтения, исключив идентификатор.

Решение:



```
> 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',
```

Практическое задание 2.3.1
Вывести список самок единорогов весом от полутонны до 700 кг, исключив вывод идентификатора.

Решение:

```
> db.unicorns.find({gender:'f', weight: {$gt:500, $lt:700}}, {_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'
  }
```

Практическое задание 2.3.2
Вывести список самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора.

Решение:

```
> db.unicorns.find({gender:'m', 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.

Решение:

```
> db.unicorns.find({vampires: {$exists: false}})
< {
    _id: ObjectId('6857fad8088eb32201c0ddde'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

Практическое задание 2.3.4
Вывести упорядоченный список имен самцов единорогов с информацией об их первом предпочтении.

Решение:

```
> db.unicorns.find({gender: 'm'}, {name: 1, loves: {$slice: 1}}).sort({name: 1})
< {
    _id: ObjectId('6857fc97088eb32201c0dddf'),
    name: 'Dunx',
    loves: [
      'grape'
    ]
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddd4'),
    name: 'Horny',
    loves: [
      'carrot'
    ]
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddda'),
    name: 'Kenny',
    loves: [
      'grape'
    ]
  }
  {
    _id: ObjectId('6857fad8088eb32201c0dddd'),
    name: 'Pilot',
    loves: [
      'apple'
    ]
  }
  {
    _id: ObjectId('6857fad8088eb32201c0dddb'),
    name: 'Raleigh',
    loves: [
      'apple'
    ]
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddd7'),
    name: 'Roooooodles',
    loves: [
      'apple'
    ]
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddd6'),
    name: 'Unicrom',
    loves: [
      'energon'
```

Практическое задание 3.1.1

Создайте коллекцию towns и выполните выборки по мэрам с party="I" и без party.

Решение:

```
> db.towns.insert({name: "Punxsutawney ",
  populatiuon: 6200,
  last_sensus: ISODate("2008-01-31"),
  famous_for: [""],
  mayor: {
     name: "Jim Wehrle"
  }});
  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"}});
  db.towns.insert({name: "Portland",
  populatiuon: 528000,
  last_sensus: ISODate("2009-07-20"),
  famous_for: ["beer", "food"],
  mayor: {
     name: "Sam Adams",
  party: "D"}});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('68581780e08091bad5e9563c')
    }
  }
> db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 1})
< {
    _id: ObjectId('68581780e08091bad5e9563b'),
    name: 'New York',
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
> db.towns.find({"mayor.party": {$exists: false}}, {name: 1, mayor: 1})
< {
    _id: ObjectId('68581780e08091bad5e9563a'),
    name: 'Punxsutawney ',
    mayor: {
      name: 'Jim Wehrle'
    }
  }
```

Практическое задание 3.1.2
Сформировать функцию для вывода списка самцов единорогов и вывести первых двух.

Решение:

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

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

Решение:

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

Практическое задание 3.2.2
Вывести список предпочтений.

Решение:

```
> 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.

Решение:

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

Практическое задание 3.3.1
Добавить самца Barny.

Решение:

```
> db.unicorns.save({name: "Barny", loves: ["grape"], weight: 340, gender: "m"})
✖ ▸ TypeError: db.unicorns.save is not a function
> db.unicorns.insertOne({name: "Barny", loves: ["grape"], weight: 340,
  gender: "m"})
< {
    acknowledged: true,
    insertedId: ObjectId('68581d4b088eb32201c0dde3')
  }
```

Практическое задание 3.3.2
Обновить Ayna: вес 800, вампиры 51.

Решение:

```
> db.unicorns.update({name: "Ayna"}, {$set: {weight: 800, vampires: 51}})
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddd9'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
```

Практическое задание 3.3.3
Обновить Raleigh: добавить redbull в loves.

Решение:

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

```
{
  _id: ObjectId('6857fad8088eb32201c0dddb'),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar',
    'redbull'
  ],
  weight: 421,
  gender: 'm',
  vampires: 7
}
```

Практическое задание 3.3.4
Увеличить количество убитых вампиров у всех самцов на 5.

Решение:

```
> 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
Убрать партию у мэра Портланда.

Решение:

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

Практическое задание 3.3.6
Обновить Pilot: добавить chocolate в loves.
Решение:

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

```
{
  _id: ObjectId('6857fad8088eb32201c0dddd'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
```

Практическое задание 3.3.7
Обновить Aurora: добавить sugar и lemon в loves.

Решение:

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

```
{
  _id: ObjectId('6857fad8088eb32201c0ddd5'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape',
    'sugar',
    'lemon'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Практическое задание 3.4.1

Удалить беспартийных мэров, очистить коллекцию, просмотреть коллекции.

Решение:

```
> db.towns.remove({"mayor.party": {$exists: false}})
< DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
< {
    acknowledged: true,
    deletedCount: 2
  }
> db.towns.find()
< {
    _id: ObjectId('68581780e08091bad5e9563b'),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [
      'status of liberty',
      'food'
    ],
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
> db.towns.remove({})
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.towns.find()
<
> show collections
< towns
  unicorns
```

Практическое задание 4.1.1

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

Решение:

```
> db.zones.insert({_id: "fst", name: "Forest", desc: "Dense community of trees"});
  db.zones.insert({_id: "cst", name: "Coast", desc: "Sea water and land meet and above spring water tides"});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': 'cst'
    }
  }
> db.unicorns.update({name: "Horny"}, {$set: {habitat: {$ref: "zones",
  $id: "cst"}}});
  db.unicorns.update({name: "Unicrom"}, {$set: {habitat: {$ref: "zones",
  $id: "fst"}}});
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name:{$in:["Horny", "Unicrom"]}})
< {
    _id: ObjectId('6857fad8088eb32201c0ddd4'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    habitat: DBRef('zones', 'cst')
  }
  {
    _id: ObjectId('6857fad8088eb32201c0ddd6'),
    name: 'Unicrom',
    loves: [
      'energon',
      'redbull'
    ],
    weight: 984,
    gender: 'm',
    vampires: 187,
    habitat: DBRef('zones', 'fst')
  }
```

Практическое задание 4.2.1
Проверьте, можно ли задать для коллекции unicorns индекс для ключа name с флагом unique.

Решение:

```
> db.unicorns.createIndex({name: 1}, {unique: true})
< name_1
```

Практическое задание 4.3.1
Получите информацию о всех индексах коллекции unicorns. Удалите все индексы, кроме индекса для идентификатора. Попытайтесь удалить индекс для идентификатора.

Решение:

```
> db.unicorns.getIndexes();
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
  ]
> db.unicorns.dropIndexes();
< {
    nIndexesWas: 2,
    msg: 'non-_id indexes dropped for collection',
    ok: 1
  }
> db.unicorns.dropIndex("_id");
⊗ ▸ MongoServerError[IndexNotFound]: index not found with name [_id]
> db.unicorns.dropIndex("_id_");
⊗ ▸ MongoServerError[InvalidOptions]: cannot drop _id index
```

Практическое задание 4.4.1

Создайте объемную коллекцию numbers, задействовав курсор. Выберите последние четыре документа. Проанализируйте план выполнения запроса. Сколько потребовалось времени на выполнение запроса? Создайте индекс для ключа value. Получите информацию о всех индексах коллекции numbers. Выполните запрос 2. Проанализируйте план выполнения запроса с установленным индексом. Сравните время выполнения запросов с индексом и без.

Решение:

```
> 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('68582ac24de3ea01f4b753c2')
    }
  }
> db.numbers.find().sort({$natural: -1}).limit(4)
< {
    _id: ObjectId('68582ac24de3ea01f4b753c2'),
    value: 99999
  }
  {
    _id: ObjectId('68582ac24de3ea01f4b753c1'),
    value: 99998
  }
  {
    _id: ObjectId('68582ac24de3ea01f4b753c0'),
    value: 99997
  }
  {
    _id: ObjectId('68582ac24de3ea01f4b753bf'),
    value: 99996
  }
```

Без индекса:

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

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 1,
  totalKeysExamined: 0,
  totalDocsExamined: 4,
  executionStages: {
    isCached: false,
    stage: 'LIMIT',
    nReturned: 4,
    executionTimeMillisEstimate: 0,
    works: 5,
    advanced: 4,
    needTime: 0,
    needYield: 0,
    saveState: 0,
    restoreState: 0,
    isEOF: 1,
    limitAmount: 4,
    inputStage: {
      stage: 'COLLSCAN',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 4,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 0,
      direction: 'backward',
      docsExamined: 4
```

С индексом:

```
> db.numbers.createIndex({value: 1})
< value_1
> db.numbers.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { value: 1 }, name: 'value_1' }
  ]
> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 4,
  executionTimeMillis: 0,
  totalKeysExamined: 0,
  totalDocsExamined: 4,
  executionStages: {
    isCached: false,
    stage: 'LIMIT',
    nReturned: 4,
    executionTimeMillisEstimate: 0,
    works: 5,
    advanced: 4,
    needTime: 0,
    needYield: 0,
    saveState: 0,
    restoreState: 0,
    isEOF: 1,
    limitAmount: 4,
    inputStage: {
      stage: 'COLLSCAN',
      nReturned: 4,
      executionTimeMillisEstimate: 0,
      works: 4,
      advanced: 4,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 0,
      direction: 'backward',
      docsExamined: 4
```

Вывод по заданию:
Без индекса поиск занял около 1 миллисекунды, когда с использованием индекса поиск занял менее 1 миллисекунды (видим из поля executionTimeMillis)

Выводы

В ходе лабораторной работы были освоены основные CRUD-операции в MongoDB, включая вставку, выборку, обновление и удаление документов. Рассмотрены методы работы с вложенными объектами, агрегацией данных, а также создание и управление индексами для оптимизации запросов. Практические задания позволили закрепить навыки составления сложных запросов, использования курсоров и ссылок между коллекциями.