

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)**

Факультет прикладной информатики

Образовательная программа Мобильные и сетевые технологии

Направление подготовки 09.03.03 Мобильные и сетевые технологии

О Т Ч Е Т

Лабораторная работа № 6.

Тема работы: «Работа с БД в СУБД MongoDB»

Обучающийся: Дементьев Антон Павлович, К3241

Поток: ПиРБД К23 1.4

Преподаватель: Говорова М.М.

Санкт-Петербург,
2025

Цель работы

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.

Программное обеспечение

Программное обеспечение: СУБД MongoDB 4+, 8.0.4 (последняя).

Выполнение работы

Практическое задание 2.1.1:

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

```
> use learn
< switched to db learn
> db.createCollection("unicorns")
< { ok: 1 }
> show collections
< unicorns
> db.unicorns.insert({name: 'Horny', loves: ['carrot', 'papaya'], weight: 600, gender: 'm', vampires: 63});
db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
db.unicorns.insert({name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80});
db.unicorns.insert({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
db.unicorns.insert({name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
db.unicorns.insert({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6836f1092f4a3d67b77e10ea')
  }
}
```

- 3) Используя второй способ, вставьте в коллекцию единорогов документ.

```
> db.unicorns.insertOne({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 784, gender: 'm', vampires: 165})
< {
  acknowledged: true,
  insertedId: ObjectId('6836f1fc93bc9c07db775db')
}
```

- 4) Проверьте содержимое коллекции с помощью метода *find*.

```
{
  _id: ObjectId('6836f1092f4a3d67b77e10ea'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  _id: ObjectId('6836f1fc93bc9c07db775db'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 784,
  gender: 'm',
  vampires: 165
}
```

Практическое задание 2.2.1:

- 1) Сформируйте запросы для вывода списков самцов и самок единорогов.

```
> db.unicorns.find({gender: 'm'}).sort({name: 1})
< {
  _id: ObjectId('6836f1fc93bcf9c07db775db'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10e0'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

Ограничьте список самок первыми тремя особями.

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e1'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10e5'),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 733,
  gender: 'f',
  vampires: 40
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10e8'),
  name: 'Leia',
  loves: [
    'apple',
    'watermelon'
  ],
  weight: 601,
  gender: 'f',
  vampires: 33
}
learn>
```

Отсортируйте списки по имени.

```
> db.unicorns.find({gender: 'f', loves: 'carrot'})
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e1'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10e4'),
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],
  weight: 550,
  gender: 'f',
  vampires: 80
}
}
```

- 2) Найдите всех самок, которые любят *carrot*.

Ограничьте этот список первой особью с помощью функций *findOne* и *limit*.

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e1'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
> db.unicorns.find({gender: "f", loves: "carrot"}).limit(1)
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e1'),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn>
```

Практическое задание 2.2.2:

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

```
> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e0'),
  name: 'Horny',
  weight: 600,
  vampires: 63
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10e2'),
  name: 'Unicrom',
  weight: 984,
  vampires: 182
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10e3'),
  name: 'Rooooooodles',
  weight: 575,
  vampires: 99
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10e6'),
  name: 'Kenny',
  weight: 690,
  vampires: 39
}
```

Практическое задание 2.2.3:

- 1) Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find().sort({$natural: -1})
< {
  _id: ObjectId('6836f1fc93bcf9c97db775db'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10ea'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10e9'),
  name: 'Bilbo'
```

Практическое задание 2.2.4:

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

```
> 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
```

Практическое задание 2.3.1:

- 1) Вывести список самок единорогов весом от 500 до 700 кг, исключив вывод идентификатора.

```
> 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,
```

Практическое задание 2.3.2:

- 1) Вывести список самцов единорогов весом от 500кг и предпочитающих *grape* и *lemon*, исключив вывод идентификатора.

```
> 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:

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

```
> db.unicorns.find({vampires: {$exists: false}})
< {
  _id: ObjectId('6836f1092f4a3d67b77e10ea'),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
```

Практическое задание 2.3.4:

- 1) Вывести список упорядоченный список имен самцов единорогов с информацией об их первом предпочтении.

```
> 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'
  ]
}
```

Практическое задание 3.1.1:

- 1) Создайте коллекцию *towns*, включающую следующие документы:

```
> db.towns.insert((name: "Punxsutawney ",
populatiuon: 6200,
last_sensu: ISODate("2008-01-31"),
famous_for: [""],
mayor: {
  name: "Jim Wehrle"
})
)
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683c45523c6a4748ccd8e882')
  }
}
> db.towns.insert((name: "New York",
populatiuon: 22200000,
last_sensu: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
  name: "Michael Bloomberg",
  party: "I"})
)
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683c45633c6a4748ccd8e883')
  }
}
```

```
> db.towns.insert((name: "Portland",
populatiuon: 528000,
last_sensu: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
  name: "Sam Adams",
  party: "D"})
)
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683c45a83c6a4748ccd8e884')
  }
}
```

- 2) Сформировать запрос, который возвращает список городов с независимыми мэрами (*party="I"*). Вывести только название города и информацию о мэре.

```
> db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 1})
< {
  _id: ObjectId('683c45633c6a4748ccd8e883'),
  name: 'New York',
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
```

- 3) Сформировать запрос, который возвращает список городов с независимыми мэрами (*party="I"*). Вывести только название города и информацию о мэре.

```
> db.towns.find({"mayor.party": {"$exists": false}}, {name: 1, mayor: 1})
< {
  _id: ObjectId('683c45523c6a4748ccd8e882'),
  name: 'Punxsutawney ',
  mayor: {
    name: 'Jim Wehrle'
  }
}
```

```
> db.towns.find()
< {
  _id: ObjectId('683c45523c6a4748ccd8e882'),
  name: 'Punxsutawney ',
  populatiuon: 6200,
  last_sensu: 2008-01-31T00:00:00.000Z,
  famous_for: [
    ''
  ],
  mayor: {
    name: 'Jim Wehrle'
  }
}
{
  _id: ObjectId('683c45633c6a4748ccd8e883'),
  name: 'New York',
  populatiuon: 22200000,
  last_sensu: 2009-07-31T00:00:00.000Z,
  famous_for: [
    'status of liberty',
    'food'
  ],
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
{
  _id: ObjectId('683c45a83c6a4748ccd8e884'),
  name: 'Portland',
  populatiuon: 528000,
  last_sensu: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
```

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*.
- 4) Содержание коллекции единорогов *unicorns*:

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

Практическое задание 3.2.1:

- 1) Вывести количество самок единорогов весом от 500 до 600 кг.

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

Практическое задание 3.2.2:

- 1) Вывести список предпочтений.

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

Практическое задание 3.2.3:

- 1) Посчитать количество особей единорогов обоих полов.

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

Практическое задание 3.3.1:

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

```
> db.unicorns.insertOne({name: "Barney", loves: ["grape"], weight: 340, gender: "m"})
< {
  acknowledged: true,
  insertedId: ObjectId('683c47733c6a4748ccd8e885')
}
learn>
```


- 2) Проверить содержимое коллекции *unicorns*.

```
> db.unicorns.find({name: "Barny"})
< {
  _id: ObjectId('683c4773c6a4748ccd9e885'),
  name: 'Barny',
  loves: [
    'grape'
  ],
  weight: 348,
  gender: 'm'
}
```

Практическое задание 3.3.2:

- 1) Для самки единорога *Ayna* внести изменения в БД: теперь ее вес 800, она убила 51 вампира.

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

- 2) Проверить содержимое коллекции *unicorns*.

```
> db.unicorns.find({name: "Ayna"})
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e5'),
  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.

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

- 2) Проверить содержимое коллекции *unicorns*.

```
> db.unicorns.find({name: "Raleigh"})
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e7'),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar',
    'redbull'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
```

Практическое задание 3.3.4:

- 1) Всем самцам единорогов увеличить количество убитых вампиров на 5.

```
> db.unicorns.update({gender: "m"}, {$inc: {vampires: 5}}, {multi: true})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
```

- 2) Проверить содержимое коллекции *unicorns*.

```
> db.unicorns.find({gender: "m"})
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e0'),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 68,
  habitat: DBRef('habitats', 'nw')
}
{
  _id: ObjectId('6836f1092f4a3d67b77e10e2'),
  name: 'Unicrom',
  loves: [
    'energon',
    'redbull'
  ],
  weight: 984,
  gender: 'm',
  vampires: 187
}
```

Практическое задание 3.3.5:

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

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

- 2) Проверить содержимое коллекции *towns*.

```
> db.towns.find({name: "Portland"})
< {
  _id: ObjectId('683c62b434c0302a04487bec'),
  name: 'Portland',
  population: 528000,
  last_census: 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.

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

- 2) Проверить содержимое коллекции unicorns.

```
> db.unicorns.find({name: "Pilot"})
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e9'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
```

Практическое задание 3.4.1:

- 1) Создайте коллекцию *towns*, включающую следующие документы:

```
> db.towns.find()
< {
  _id: ObjectId('683c679834c9302a04487bed'),
  name: 'Punxsutawney ',
  population: 6200,
  last_sensus: 2008-01-31T00:00:00.000Z,
  famous_for: [
    'phil the groundhog'
  ],
  mayor: {
    name: 'Jim Wehrle'
  }
},
{
  _id: ObjectId('683c67a634c9302a04487bee'),
  name: 'New York',
  population: 22200000,
  last_sensus: 2009-07-31T00:00:00.000Z,
  famous_for: [
    'status of liberty',
    'food'
  ],
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
},
{
  _id: ObjectId('683c67b034c9302a04487bef'),
  name: 'Portland',
  population: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams',
    party: 'D'
  }
}
```

- 2) Удалите документы с беспартийными мэрами.

```
> db.towns.remove({"mayor.party": {$exists: false}})
< {
  acknowledged: true,
  deletedCount: 1
}
learn>|
```

- 3) Проверьте содержание коллекции.

```
> db.towns.find()
< {
  _id: ObjectId('683c67a634c0302a04487bee'),
  name: 'New York',
  population: 22200000,
  last_sensus: 2009-07-31T00:00:00.000Z,
  famous_for: [
    'status of liberty',
    'food'
  ],
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
{
  _id: ObjectId('683c67b034c0302a04487bef'),
  name: 'Portland',
  population: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams',
    party: 'D'
  }
}
learn>|
```

- 4) Очистите коллекцию.

```
> db.towns.remove({})
< {
  acknowledged: true,
  deletedCount: 2
}
learn>|
```

- 5) Просмотрите список доступных коллекций.

```
> show collections
< towns
unicorns
> db.towns.find()
<
```

Практическое задание 4.1.1:

- 1) Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.

```
> db.habitats.insert({_id: "nw", name: "Northwest", desc: "Forests and rivers"});
< {
  acknowledged: true,
  insertedIds: {
    '0': 'nw'
  }
}
> db.habitats.insert({_id: "desert", name: "Desert", desc: "Hot and sandy"});
< {
  acknowledged: true,
  insertedIds: {
    '0': 'desert'
  }
}
```

- 2) Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания.

```
> db.unicorns.update({name: "Pilot"}, {$set: {habitat: {$ref: "habitats", $id: "desert"}}});
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.update({name: "Dunx"}, {$set: {habitat: {$ref: "habitats", $id: "nw"}}});
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

- 3) Проверьте содержание коллекции единорогов.

```
> db.unicorns.find({name: "Dunx"})
< {
  _id: ObjectId('6836f1fc93bcf9c07db775db'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 170,
  habitat: DBRef('habitats', 'nw')
}
> db.unicorns.find({name: "Pilot"})
< {
  _id: ObjectId('6836f1092f4a3d67b77e10e9'),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59,
  habitat: DBRef('habitats', 'desert')
}
```

Практическое задание 4.2.1:

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

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

Практическое задание 4.3.1:

- 1) Получите информацию о всех индексах коллекции *unicorns*.

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

- 2) Удалите все индексы, кроме индекса для идентификатора.

```
> db.unicorns.dropIndexes();
< {
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
```

- 3) Попробуйте удалить индекс для идентификатора.

```
> db.unicorns.dropIndex("_id");
MongoServerError[InvalidOptions]: 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})}```
```

```
> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
< {
 acknowledged: true,
 insertedIds: {
 '0': ObjectId('683c4bf53c6a4748ccdbd93c')
 }
}
```

- 2) Выберите последних четыре документа.

```
> db.numbers.find().sort({$natural: -1}).limit(4)
< {
 _id: ObjectId('683c4bf53c6a4748ccdbd93c'),
 value: 99999
}
{
 _id: ObjectId('683c4bf53c6a4748ccdbd93b'),
 value: 99998
}
{
 _id: ObjectId('683c4bf53c6a4748ccdbd93a'),
 value: 99997
}
{
 _id: ObjectId('683c4bf53c6a4748ccdbd939'),
 value: 99996
}
```

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

```
> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
< {
 explainVersion: '1',
 queryPlanner: {
 namespace: 'learn.numbers',
 parsedQuery: {},
 indexFilterSet: false,
 queryHash: '8F2383EE',
 planCacheShapeHash: '8F2383EE',
 planCacheKey: '7DF350EE',
 optimizationTimeMillis: 0,
 maxIndexedOrSolutionsReached: false,
 maxIndexedAndSolutionsReached: false,
 maxScansToExplodeReached: false,
 prunedSimilarIndexes: false,
 winningPlan: {
 isCached: false,
 stage: 'LIMIT',
 limitAmount: 4,
 inputStage: {
 stage: 'COLLSCAN',
 direction: 'backward'
 }
 },
 rejectedPlans: []
 },
 executionStats: {
 executionSuccess: true,
 nReturned: 4,
 executionTimeMillis: 0,
 totalKeysExamined: 0,
 totalDocsExamined: 4,
 executionStages: {
 isCached: false,
 stage: 'LIMIT',
 nReturned: 4,
 executionTimeMillisEstimate: 0,

```

- 4) Создайте индекс для ключа `value`.  
5) Получите информацию о всех индексах коллекции `numbers`.


```
> 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' }
]
```

- 6) Выполните запрос 2.

```
> db.numbers.find().sort({$natural: -1}).limit(4)
< {
 _id: ObjectId('683c6d0434c0302a044a028f'),
 value: 99999
}
{
 _id: ObjectId('683c6d0434c0302a044a028e'),
 value: 99998
}
{
 _id: ObjectId('683c6d0434c0302a044a028d'),
 value: 99997
}
{
 _id: ObjectId('683c6d0434c0302a044a028c'),
 value: 99996
}
```

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

```
> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
< {
 explainVersion: '1',
 queryPlanner: {
 namespace: 'learn.numbers',
 parsedQuery: {},
 indexFilterSet: false,
 queryHash: '0F2383EE',
 planCacheShapeHash: '8F2383EE',
 planCacheKey: '7DF350EE',
 optimizationTimeMillis: 0,
 maxIndexedOrSolutionsReached: false,
 maxIndexedAndSolutionsReached: false,
 maxScansToExplodeReached: false,
 prunedSimilarIndexes: false,
 winningPlan: {
 isCached: false,
 stage: 'LIMIT',
 limitAmount: 4,
 inputStage: {
 stage: 'COLLSCAN',
 direction: 'backward'
 }
 },
 rejectedPlans: []
 },
 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
 }
 }
}
```



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

**Ответ:** Запрос с использованием индекса выполняется значительно быстрее (около 0 мс) по сравнению с запросом без индекса (1 мс), так как индекс позволяет избежать полного просмотра коллекции. Запрос с использованием индекса намного эффективнее.



## Контрольные вопросы:

### Пункт 3

#### 1) Как используется оператор точка?

Оператор точка (.) используется для доступа к полям вложенных документов и элементам массивов.

#### 2) Как можно использовать курсор?

Курсор позволяет итерироваться по результатам запроса.

Его можно использовать для:

- Постепенной обработки больших наборов данных.
- Применения методов, таких как *.limit()*, *.skip()*, *.sort()*.
- Преобразования результатов с помощью *.map()* или *.forEach()*.

#### 3) Какие возможности агрегирования данных существуют в MongoDB?

MongoDB предоставляет мощный Aggregation Framework, включающий:

- Этапы пайплайна (*\$match*, *\$group*, *\$sort*, *\$project*, *\$lookup* и др.).
- Операторы агрегации (*\$sum*, *\$avg*, *\$max*, *\$min*, *\$push* и др.).
- Вычисления и преобразования данных (арифметика, строковые операции, работа с датами).
- Джойны между коллекциями (*\$lookup*).

#### 4) Какая из функций *save* или *update* более детально позволит настроить редактирование документов коллекции?

Метод *update* (или *updateOne/updateMany*) предоставляет более гибкие настройки, так как позволяет:

- Точечно изменять поля с помощью операторов (*\$set*, *\$unset*, *\$inc*).
- Использовать условия для выбора документов.
- Применять сложные модификации с агрегационными операциями (*\$addToSet*, *\$pull*).
- Метод *save* просто перезаписывает документ (если *\_id* существует) или вставляет новый (если *\_id* нет).

#### 5) Как происходит удаление документов из коллекции по умолчанию?

По умолчанию методы *deleteOne()* и *deleteMany()* физически удаляют документы из коллекции.

## Пункт 4

### 1) Назовите способы связывания коллекций в MongoDB.

- Вложение документов (Embedding) – хранение связанных данных внутри одного документа.
- Ссылки (Reference) – хранение *ObjectId* и использование *\$lookup* для джойнов.
- Денормализация – дублирование данных для ускорения чтения.

### 2) Сколько индексов можно установить на одну коллекцию в БД MongoDB?

В MongoDB нет жесткого ограничения на количество индексов, но рекомендуется не более 64 индексов на коллекцию.

### 3) Как получить информацию о всех индексах базы данных MongoDB?

```
db.collection.getIndexes()
```

**Вывод:**

В процессе выполнения лабораторной работы я освоил основные принципы работы с MongoDB. Были изучены базовые операции по управлению данными: создание, чтение, обновление и удаление документов (CRUD-операции). Особое внимание уделялось работе со сложными структурами данных, включая вложенные документы и массивы.

Полученные знания дают мне понимание того, как эффективно использовать MongoDB в реальных проектах. Я научился выбирать оптимальные подходы к хранению и обработке данных в зависимости от конкретных задач. Эти навыки будут полезны при разработке современных приложений, работающих с большими объемами информации.