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.

Выполнение:

2.1.1)

1. Создать базу данных learn.
2. Заполнить коллекцию единорогов unicorns
3. Использовать второй способ, вставить в коллекцию единорогов документ

```
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('6837ab8a630eed105129b744') }
}
```

=

```
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
db.unicorns.insertMany  db.unicorns.insertOne

learn> db.unicorns.insert
db.unicorns.insertMany  db.unicorns.insertOne

learn> db.unicorns.insertOne(document)
{
  acknowledged: true,
  insertedId: ObjectId('6837ad71630eed105129b745')
}
```

2.2.1)

Сформировать запросы для вывода списков самцов и самок единорогов. Ограничить список самок первыми тремя особями. Отсортировать списки по имени.

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

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('6837ab8a630eed105129b73a'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('6837ab8a630eed105129b73b'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6837ab8a630eed105129b73c'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('6837ab8a630eed105129b73d'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6837ab8a630eed105129b73e'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
```

```
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
}
```

2.2.3) Вывести список единорогов в обратном порядке добавления.

```
learn> db.unicorns.find().sort({$natural : -1})
[
  {
    _id: ObjectId('6837ad71630eed105129b745'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('6837ab8a630eed105129b744'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId('6837ab8a630eed105129b743'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId('6837ab8a630eed105129b742'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('6837ab8a630eed105129b741'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId('6837ab8a630eed105129b740'),
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId('6837ab8a630eed105129b73f'),
```

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

```
learn> db.unicorns.find({},{_id : false, name : true, loves : {$slice : 1}})
[
  { name: 'Horny', loves: [ 'carrot' ] },
  { name: 'Aurora', loves: [ 'carrot' ] },
  { name: 'Unicrom', loves: [ 'energon' ] },
  { name: 'Roooooodles', loves: [ 'apple' ] },
  { name: 'Solnara', loves: [ 'apple' ] },
  { name: 'Ayna', loves: [ 'strawberry' ] },
  { name: 'Kenny', loves: [ 'grape' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Leia', loves: [ 'apple' ] },
  { name: 'Pilot', loves: [ 'apple' ] },
  { name: 'Nimue', loves: [ 'grape' ] },
  { name: 'Dunx', loves: [ 'grape' ] }
]
```

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

```
learn> db.unicorns.find({weight : {$gte : 500, $lte: 700}, gender : 'f'}, {_id : false, name : 1, weight : 1})
[
  { name: 'Solnara', weight: 550 },
  { name: 'Leia', weight: 601 },
  { name: 'Nimue', weight: 540 }
]
```

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:0, name:1, weight:1,loves:1})
[ { name: 'Kenny', loves: [ 'grape', 'lemon' ], weight: 690 } ]
learn>
```

2.3.3) Найти всех единорогов, не имеющих ключ vampires.

```
learn> db.unicorns.find({vampires : {$exists : false}})
[
  {
    _id: ObjectId('6837ab8a630eed105129b744'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

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

```
learn> db.unicorns.find({gender : 'm'}, {_id : 0, loves : {$slice : 1}, name : 1}).sort({name:1})
[
  { name: 'Dunx', loves: [ 'grape' ] },
  { name: 'Horny', loves: [ 'carrot' ] },
  { name: 'Kenny', loves: [ 'grape' ] },
  { name: 'Pilot', loves: [ 'apple' ] },
  { name: 'Raleigh', loves: [ 'apple' ] },
  { name: 'Roooooodles', loves: [ 'apple' ] },
  { name: 'Unicrom', loves: [ 'energon' ] }
]
```

3.1.1)

1. Создать коллекцию towns, сформировать запрос, который возвращает список городов с независимыми мэрами (party="I").
2. 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 мэре.

```
learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     populatiuon: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: [],
...     mayor: {
...       name: "Jim Wehrle"
...     }
...   },
...   {
...     name: "New York",
...     populatiuon: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["status of liberty", "food"],
...     mayor: {
...       name: "Michael Bloomberg",
...       party: "I"
...     }
...   },
...   {
...     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('6837bd73630eed105129b746'),
    '1': ObjectId('6837bd73630eed105129b747'),
    '2': ObjectId('6837bd73630eed105129b748')
  }
}
```

```
learn> db.towns.find({"mayor.party" : "I"}, {_id : 0 , name : 1, mayor : 1})
[
  {
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
learn> db.towns.find({"mayor.party" : {$exists: false}}, {_id : 0 , name : 1, mayor : 1})
[ { name: 'Punxsutawney', mayor: { name: 'Jim Wehrle' } } ]
```

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> var cursor = db.unicorns.find({gender : 'm'}).sort({name : 1}).limit(2)

[learn> cursor.forEach(function(obj){
[... print(obj.name);
[... })
 Dunx
 Horny
```

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

```
[learn> db.unicorns.find({gender : 'f', weight : {$gte : 500, $lte : 600}}).count()
 (node:40375) [MONGODB DRIVER] Warning: cursor.count is deprecated and will be removed
 (Use `node --trace-warnings ...` to show where the warning was created)
 2
```

3.2.2) Вывести список предпочтений

3.2.3) Посчитать количество особей единорогов обоих полов.

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

3.3.1) Выполнить команду:

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

3.3.2) Для самки единорога Ayna внести изменения в БД: теперь ее вес 800, она убила 51 вапмира. Проверить содержимое коллекции unicorns.

```
[learn> db.unicorns.updateOne({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('6837ab8a630eed105129b73f'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

3.3.3) Для самца единорога Raleigh внести изменения в БД: теперь он любит рэдбул. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update({name : "Raleigh"}, {$set : {loves : "Redbull"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({name : "Raleigh"})
[
  {
    _id: ObjectId('6837ab8a630eed105129b741'),
    name: 'Raleigh',
    loves: 'Redbull',
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

3.3.4) Всем самцам единорогов увеличить количество убитых вапмиров на 5. Проверить содержимое коллекции unicorns.

```
learn> db.unicorns.update({gender : 'm'}, {$inc : {vampires:5}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.u
db.updateUser  db.updateRole  db.unicorns

learn> db.unicorns.find()
[
  {
    _id: ObjectId('6837ab8a630eed105129b73a'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 68
  },
  {
    _id: ObjectId('6837ab8a630eed105129b73b'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('6837ab8a630eed105129b73c'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('6837ab8a630eed105129b73d'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('6837ab8a630eed105129b73e'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
```

3.3.5) Изменить информацию о городе Портланд: мэр этого города теперь беспартийный. Проверить содержимое коллекции towns.

3.3.6) Изменить информацию о самце единорога Pilot: теперь он любит и шоколад. Проверить содержимое коллекции unicorns.

```
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('6837ab8a630eed105129b743'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
]
```

3.3.7) Изменить информацию о самке единорога Aurora: теперь она любит еще и сахар, и лимоны. Проверить содержимое коллекции unicorns.

```
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('6837ab8a630eed105129b73b'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

3.4.1)

1. Удалить документы с беспартийными мэрами.
2. Проверить содержание коллекции
3. Очистить коллекцию
4. Посмотреть список доступных коллекций.

```
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('6837bd73630eed105129b747'),
    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
towns
unicorns
learn>
```

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.habitats.insertMany([
...   {
...     _id: 'forest',
...     name: 'Enchanted Forest',
...     description: 'A lush, magical forest filled with glowing flora and sparkling rivers.'
...   },
...   {
...     _id: 'mountain',
...     name: 'Crystal Mountains',
...     description: 'Snow-covered peaks that shine with ancient crystals.'
...   },
...   {
...     _id: 'valley',
...     name: 'Rainbow Valley',
...     description: 'A peaceful valley where rainbows are constantly visible and the grass sings.'
...   }
... ]);
[...
{
  acknowledged: true,
  insertedIds: { '0': 'forest', '1': 'mountain', '2': 'valley' }
}
```

```
learn> db.unicorns.updateOne(
...   { name: 'Pilot' },
...   { $set: { habitat: 'forest' } }
... );
...
... db.unicorns.updateOne(
...   { name: 'Leia' },
...   { $set: { habitat: 'valley' } }
... );
...
... db.unicorns.updateOne(
...   { name: 'Unicorn' },
...   { $set: { habitat: 'mountain' } }
... );
[...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

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

```
learn> db.unicorns.ensureIndex({"name" : 1}, {"unique" : true})
MongoServerError[DuplicateKey]: Index build failed: 764ae27b-769e-4085-a3cf-a2709448afc0: Collection learn.unicorns ( 513a6503-f3e9-4af6-b40a-4e3ef0c7d910 ) :: caused by :: E11000 duplicate key error collection: learn.unicorns index: name_1 dup key: { name: "Aurora" }
learn>
```

4.3.1)

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

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

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

```
learn> db.unicorns.dropIndex("_id_");
[...
MongoServerError[InvalidOptions]: cannot drop _id index
learn>
```

4.4.1)

1. Создайте объемную коллекцию numbers, задействовав курсор
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 numbers.
6. Выполните запрос 2.
7. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?
8. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?

```
learn> for (let 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('683856abbd2ba112798181ed') }
}
learn>
```

```
learn> db.numbers.find().sort({ value: -1 }).limit(4).explain("executionStats");
...
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'BA27D965',
    planCacheShapeHash: 'BA27D965',
    planCacheKey: '7A892B81',
    optimizationTimeMillis: 13,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'SORT',
      sortPattern: { value: -1 },
      memLimit: 104857600,
      limitAmount: 4,
      type: 'simple',
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 113,
    totalKeysExamined: 0,
    totalDocsExamined: 277423,
    executionStages: {
      isCached: false,
      stage: 'SORT',
      nReturned: 4,
      executionTimeMillisEstimate: 95,
      works: 277429,
      advanced: 4,
      needTime: 277424,
      needYield: 0,
      saveState: 2,
      restoreState: 2,
      isEOF: 1,
      sortPattern: { value: -1 },
      memLimit: 104857600,
      limitAmount: 4,
      type: 'simple',
      totalDataSizeSorted: 260,
      usedDisk: false,
      spills: 0,
      spilledDataStorageSize: 0,
      inputStage: {
        stage: 'COLLSCAN',
        nReturned: 277423,
        executionTimeMillisEstimate: 5,
        works: 277424,
        advanced: 277423,
        needTime: 0,
        needYield: 0,
        saveState: 2,
        restoreState: 2,
        isEOF: 1,
        direction: 'forward',
        docsExamined: 277423
      }
    }
  },
```

```
learn> db.numbers.createIndex({ value: 1 });
[...
value_1
learn> db.numbers.getIndexes();
[...
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
learn> db.numbers.find().sort({ value: -1 }).limit(4).explain("executionStats");
[...
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'BA27D965',
    planCacheShapeHash: 'BA27D965',
    planCacheKey: '7A892B81',
    optimizationTimeMillis: 5,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: {
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { value: 1 },
          indexName: 'value_1',
          isMultiKey: false,
          multiKeyPaths: { value: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'backward',
          indexBounds: { value: [ '[MaxKey, MinKey]' ] }
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 15,
    totalKeysExamined: 4,
    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: 'FETCH',
        nReturned: 4,
        executionTimeMillisEstimate: 0,
        works: 4,
        advanced: 4,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 0,
        docsExamined: 4,
        alreadyHasObj: 0,
        inputStage: {
          stage: 'IXSCAN',
          nReturned: 4,
          executionTimeMillisEstimate: 0,
          works: 4,
          advanced: 4,
          needTime: 0,
          needYield: 0,
          saveState: 0,
          restoreState: 0,
          isEOF: 0,
          keyPattern: { value: 1 },
          indexName: 'value_1',
          isMultiKey: false,
          multiKeyPaths: { value: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'backward',
          indexBounds: { value: [ '[MaxKey, MinKey]' ] },
          keysExamined: 4,
          seeks: 1,
          dupsTested: 0,
          dupsDropped: 0
        }
      }
    }
  },
```

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.