

Бд лаб 5

Оглавление

Оглавление

8.1

8.1.2

8.1.3

8.1.4

8.1.6

8.1.7

8.1.8

8.1.9

8.2.1

8.2.2

8.2.3

8.2.4

8.2.5

8.2.8

8.2.9

8.2.10

8.2.11,12

8.2.13

8.3.1

8.3.2

8.3.3

8.3.4

8.1

добавление

1. *Создайте базу данных `learn`.*
2. *Заполните коллекцию единорогов `unicorns`:*
3. *Используя второй способ, вставьте в коллекцию единорогов документ:*
4. *Проверьте содержимое коллекции с помощью метода `find`.*

правда я назвал ее - mydb

```
    insertedIds: {
      '0': ObjectId("649bf855a1a023d4fdae4a2")
    }
  }
> db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("649bf85ba1a023d4fdae4a3")
  }
}
> db.unicorns.insert({name: 'Roocoooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("649bf86ea1a023d4fdae4a4")
  }
}
> db.unicorns.insert({name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80});
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("649bf873a1a023d4fdae4a5")
  }
}
> db.unicorns.insert({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("649bf87aa1a023d4fdae4a6")
  }
}
> db.unicorns.insert({name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
< {
  acknowledged: true,
  insertedIds: {
```

добавление вторым способом

```
> 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("649bf963a1a023d4fdae4ac")
  }
}
```

проверяем

```
> db.unicorns.find()
< {
  _id: ObjectId("649bf7aba1a023d4fdaee4a1"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId("649bf855a1a023d4fdaee4a2"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId("649bf85ba1a023d4fdaee4a3"),
  name: 'Unicrom',
  loves: [
```

8.1.2

1. Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.

```

> db.unicorns.find({gender: 'm'}).sort({name:1})
< {
  _id: ObjectId("649bf963a1a023d4fdaee4ac"),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId("649bf7aba1a023d4fdaee4a1"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ]
}

```

2. Найдите всех самок, которые любят carrot. Ограничьте этот список первой особью с помощью функций *findOne* и *limit*.

```
}
> db.unicorns.find({gender: 'f'}).limit(3).sort({name: 1})
< {
  _id: ObjectId("649bf855a1a023d4fdaee4a2"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId("649bf87aa1a023d4fdaee4a6"),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
```

```

> db.unicorns.find({gender: 'f', loves: 'carrot'})
< {
  _id: ObjectId("649bf855a1a023d4fdaee4a2"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId("649bf873a1a023d4fdaee4a5"),
  name: 'Solnara',
  loves: [
    'apple',
    'carrot',
    'chocolate'
  ],

```

8.1.3

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

```
> db.unicorns.find({gender: 'm'}, {likes: 0, gender: 0})
< {
  _id: ObjectId("649bf7aba1a023d4fdaee4a1"),
  name: 'Horny',
  weight: 600,
  vampires: 63
}
{
  _id: ObjectId("649bf85ba1a023d4fdaee4a3"),
  name: 'Unicrom',
  weight: 984,
  vampires: 182
}
{
  _id: ObjectId("649bf86ea1a023d4fdaee4a4"),
  name: 'Rooooooodles',
  weight: 575,
  vampires: 99
}
{
```

8.1.4

Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find().sort({$natural: -1})
< {
  _id: ObjectId("649bf963a1a023d4fdaee4ac"),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId("649bf89fa1a023d4fdaee4ab"),
  name: 'Nimue',
  loves: [
```



```

        'apple',
        'carrot',
        'chocolate'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
}
{
    _id: ObjectId("649bf86ea1a023d4fdaee4a4"),
    name: 'Rooooooodles',
    loves: [
        'apple'
    ],
    weight: 575,
    gender: 'm',
    vampires: 99
}
{
    _id: ObjectId("649bf85ba1a023d4fdaee4a3"),
    name: 'Unicrom',
    loves: [
        'energon',
        'redbull'
    ],
    weight: 984,
    gender: 'm',
    vampires: 182
}
{
    _id: ObjectId("649bf855a1a023d4fdaee4a2"),
    name: 'Aurora',
    loves: [
        'carrot',

```

8.1.6

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

```
> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 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'
}
```

8.1.7

Вывести список самцов единорогов весом от полутонны 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
}
mydb>
```

8.1.8

Найти всех единорогов, не имеющих ключ vampires.

```
> db.unicorns.find({vampires: {$exists: false}})
< {
  _id: ObjectId("649bf89fa1a023d4fdaee4ab"),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
mydb>
```

8.1.9

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'
  ]
}
{
  name: 'Raleigh',
  loves: [
    'apple'
  ]
}
{
  name: 'Rooooooodles',
  loves: [
    'apple'
  ]
}
```

8.2.1

1. Создайте коллекцию *towns*, включающую следующие документы:

```

    }
> 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("649c0063a1a023d4fdaee4ae")
  }
}
> db.towns.insert({name: "Portland",
  populatiuon: 528000,
  last_sensus: ISODate("2009-07-20"),
  famous_for: ["beer", "food"],
  mayor: {
    name: "Sam Adams",
    party: "D"}}
)
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("649c0073a1a023d4fdaee4af")
  }
}

```

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: 0})
< {
  name: 'New York',
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
mydb> |

```

3. Сформировать запрос, который возвращает список беспартийных мэров (party отсутствует). Вывести только название города и информацию о мэре.

```

> db.towns.find({"mayor.party": {$exists: false}}, {name: 1, mayor: 1, _id: 0});
< {
  name: 'Punxsutawney ',
  mayor: {
    name: 'Jim Wehrle'
  }
}
mydb> |

```

8.2.2

Сформировать функцию для вывода списка самцов единорогов.

Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.

Вывести результат, используя *forEach*.

Содержание коллекции единорогов *unicorns*:

```

}
> var cursor = db.unicorns.find({gender: 'm'})
> var cursor = db.unicorns.find({gender: 'm'}); null
< null
> cursor.limit(2).sort({name: 1 }); null;
< null
> cursor.forEach(function(k) {print(obj);})

```

```
> cursor.forEach(function(k) {print(k);})
< {
  _id: ObjectId("649bf7aba1a023d4fdaee4a1"),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
```

8.2.3

Вывести количество самок единорогов весом от полутонны до 600 кг.

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

8.2.4

Вывести список предпочтений.

```
< 2
> db.unicorns.distinct("loves")
< [
  'apple',      'carrot',
  'chocolate', 'energon',
  'grape',      'lemon',
  'papaya',     'redbull',
  'strawberry', 'sugar',
  'watermelon'
]
```

8.2.5

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

8.2.8

1. Для самца единорога *Raleigh* внести изменения в БД: теперь он любит рэдбул.
2. Проверить содержимое коллекции *unicorns*.

```
}
> db.unicorns.find({name: "Raleigh"})
< {
  _id: ObjectId("649bf88fa1a023d4fdaee4a8"),
  name: 'Raleigh',
  loves: [
    'apple',
    'sugar'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
mydb> |
```

8.2.9

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

```
}
> db.unicorns.updateMany({gender: 'm'}, {$inc : {vampires: 5}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
> db.unicorns.find({gender: 'm'})
< {
  _id: ObjectId("649bf7aba1a023d4fdaee4a1"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 68
}
```

8.2.10

1. Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.
2. Проверить содержимое коллекции *towns*.

```

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

```

```

}
> db.towns.find({name: "Portland"})
< {
  _id: ObjectId("649c0073a1a023d4fdaee4af"),
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams'
  }
}
mydb> |

```

8.2.11,12

1. Изменить информацию о самце единорога *Pilot*: теперь он любит и шоколад.
 2. Проверить содержимое коллекции *unicorns*.
-
1. Изменить информацию о самке единорога *Aurora*: теперь она любит еще и сахар, и лимоны.
 2. Проверить содержимое коллекции *unicorns*.

```

> db.unicorns.updateOne({name: "Pilot", gender: 'm'}, {$push: {loves: "chocolate"}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.find({name: "Pilot"})
< {
  _id: ObjectId("649bf899a1a023d4fdaee4aa"),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
> db.unicorns.find({name: "Aurora"})
< {
  _id: ObjectId("649bf855a1a023d4fdaee4a2"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
mydb> |

```

8.2.13

1. Создайте коллекцию *towns*, включающую следующие документы:
2. Удалите документы с беспартийными мэрами.

3. Проверьте содержание коллекции.
4. Очистите коллекцию.
5. Просмотрите список доступных коллекций.

```

> db.towns.find()
< {
  _id: ObjectId("649c0038a1a023d4fdaee4ad"),
  name: 'Punxsutawney ',
  populatiuon: 6200,
  last_sensus: 2008-01-31T00:00:00.000Z,
  famous_for: [
    ''
  ],
  mayor: {
    name: 'Jim Wehrle'
  }
}
{
  _id: ObjectId("649c0063a1a023d4fdaee4ae"),
  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'
  }
}
{
  _id: ObjectId("649c0073a1a023d4fdaee4af"),
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [

```

```

    ],
    mayor: {
      name: 'Jim Wehrle'
    }
  }
{
  _id: ObjectId("649c0063a1a023d4fdaee4ae"),
  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'
  }
}
{
  _id: ObjectId("649c0073a1a023d4fdaee4af"),
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams'
  }
}
mydb>

```

8.3.1

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.

```
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("649c1c78a1a023d4fdaee4b0"),
    '1': ObjectId("649c1c78a1a023d4fdaee4b1")
  }
}
> db.locations.find()
< {
  _id: ObjectId("649c1c78a1a023d4fdaee4b0"),
  id: 'md',
  name: 'meadow',
  description: 'the best meadow'
}
{
  _id: ObjectId("649c1c78a1a023d4fdaee4b1"),
  id: 'fr',
  name: 'forest',
  description: 'best fortest'
}
mydb> |
```

```

> db.unicorns.update({name: 'Dunx'}, {$set: {locations: {$ref: "locations", $id: "fr"}}})
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.update({name: 'Ayna'}, {$set: {locations: {$ref: "locations", $id: "md"}}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

```

8.3.2

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

```

}
> db.unicorns.ensureIndex({"name": 1}, {"unique": true})
< [ 'name_1' ]
> db.unicorns.insert({name: 'Kenny', loves: ['redbull', 'sugar'], weight: 500, gender: 'm',
* ► MongoBulkWriteError: E11000 duplicate key error collection: mydb.unicorns index: name_1
> db.unicorns.insert({name: 'Kenny', loves: ['fish and chips'], weight: 500, gender: 'm', var
* ▼ MongoBulkWriteError: E11000 duplicate key error collection: mydb.unicorns index: name_1
Result:
{
  insertedCount: 0,
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {},
  insertedIds: {
    '0': {}
  }
}
}

```

8.3.3

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

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

✖ **MongoServerError:** cannot drop _id index

```
at Connection.onMessage (C:\Users\guann\AppData\Local\MongoDBCompass\app-1.38.0\resources\app\src\connection\connection.js:100:11)
at MessageStream.<anonymous> (C:\Users\guann\AppData\Local\MongoDBCompass\app-1.38.0\resources\app\src\connection\connection.js:100:11)
at MessageStream.emit (node:events:513:28)
at p (C:\Users\guann\AppData\Local\MongoDBCompass\app-1.38.0\resources\app\src\connection\connection.js:100:11)
at MessageStream._write (C:\Users\guann\AppData\Local\MongoDBCompass\app-1.38.0\resources\app\src\connection\connection.js:100:11)
at writeOrBuffer (node:internal/streams/writable:392:12)
at _write (node:internal/streams/writable:333:10)
at Writable.write (node:internal/streams/writable:337:10)
at Socket.ondata (node:internal/streams/readable:766:22)
at Socket.emit (node:events:513:28)
```

8.3.4

1. Создайте объемную коллекцию *numbers*, задействовав курсор:

```
for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
```
1. Выберите последних четыре документа.
2. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра *executionTimeMillis*)

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

```
✖ ▶ Error: clone(t={}){const r=t.loc||{};return e({loc:new Position("line"in r?r
> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
✖ ▶ Error: Async script execution was interrupted
> db.numbers.find().sort({$natural: -1}).limit(4)
< {
  _id: ObjectId("649c4147a1a023d4fdaf4a65"),
  value: 26033
}
{
  _id: ObjectId("649c4147a1a023d4fdaf4a64"),
  value: 26032
}
{
  _id: ObjectId("649c4147a1a023d4fdaf4a63"),
  value: 26031
}
{
  _id: ObjectId("649c4147a1a023d4fdaf4a62"),
  value: 26030
}
> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
✖ ▶ Error: clone(t={}){const r=t.loc||{};return e({loc:new Position("line"in r?r
```

```

✖ Error: clone(t={}){const r=t.loc||{};return e({loc:new Position("line" in r?r.lin
> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'mydb.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: {
        stage: 'COLLSCAN',
        direction: 'backward'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    -Returned: 4
  }
}

```

```
executionSuccess: true,  
nReturned: 4,  
executionTimeMillis: 1,  
totalKeysExamined: 0,  
totalDocsExamined: 4,  
executionStages: {  
  stage: 'LIMIT',  
  nReturned: 4,  
  executionTimeMillisEstimate: 0,  
  works: 6,  
  advanced: 4,  
  needTime: 1,  
  needYield: 0,  
  saveState: 0,  
  restoreState: 0,  
  isEOF: 1,  
  limitAmount: 4,  
  inputStage: {  
    stage: 'COLLSCAN',  
    nReturned: 4,  
    executionTimeMillisEstimate: 0,  
    works: 5,  
    advanced: 4,  
    needTime: 1,  
    needYield: 0,  
    saveState: 0,
```

```

OK: 1
}
db.numbers.ensureIndex({"value": -1})
[ 'value_-1' ]
db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: -1 }, name: 'value_-1' }
]
db.numbers.find().sort({$natural: -1}).limit(4)
{
  _id: ObjectId("649c4253a1a023d4fdafc299"),
  value: 56805
}
{
  _id: ObjectId("649c4253a1a023d4fdafc298"),
  value: 56804
}
{
  _id: ObjectId("649c4253a1a023d4fdafc297"),
  value: 56803
}
{
  _id: ObjectId("649c4253a1a023d4fdafc296"),
  value: 56802
}

```

```

> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'mydb.numbers',
    indexFilterSet: false,
    parsedQuery: {},
    queryHash: '17830885',
    planCacheKey: '17830885',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'LIMIT',
      limitAmount: 4,
      inputStage: {
        stage: 'COLLSCAN',
        direction: 'backward'
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,

```

```

    filter: {},
    sort: {
      '$natural': -1
    },
    limit: 4,
    '$db': 'mydb'
  },
  serverInfo: {
    host: 'LAPTOP-AAK0TN6T',
    port: 27017,
    version: '6.0.6',
    gitVersion: '26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
  },
  ok: 1
}

```

Эффективней по индексам