ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет инфокоммуникационных технологий
Дисциплина:
«Базы данных»
ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

«Работа с БД в СУБД MongoDB»

Выполнил:
студент группы K32391
Кравченко Богдан Игоревич

(подпись)

Проверил:
Говорова Марина Михайловна

(отметка о выполнении)

(подпись)

Санкт-Петербург
2022 г.

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.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4+, 6.0.6 (текущая).

БАЗА ДАННЫХ ДОКУМЕНТОВ

Практическое задание 8.1.1:

Создайте базу данных learn.
Заполните коллекцию единорогов unicorns:

1. *use learn*
2. OK
3. *doc = {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}*

db.unicorns.insert(doc)

4. db.unicorns.find()

```
unicorns> db.unicorns.find()
[
  {
    _id: ObjectId("651c43be938a20962c25cb4b"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb4c"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb4d"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb4e"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb4f"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb50"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733
```

Практическое задание 8.1.2:

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.

1.

a. db.unicorns.find({gender : “m”})

```
unicorns> db.unicorns.find({gender : "m"})
[
  {
    _id: ObjectId("651c43be938a20962c25cb4b"),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb4d"),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb4e"),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
```

b. db.unicorns.find({gender : "f"}).limit(3).sort({name : 1})

```
unicorns> db.unicorns.find({gender : "f"}).limit(3).sort({name : 1})
[
  {
    _id: ObjectId("652e560c49deea222ad5d1cc"),
    name: 'Aurora',
    dob: ISODate("1991-01-24T10:00:00.000Z"),
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb4c"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb50"),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
]
```

2. .

a. db.unicorns.findOne({gender:"f", loves:"carrot"})

```
unicorns> db.unicorns.findOne({gender:"f", loves:"carrot"})
{
  _id: ObjectId("651c43bf938a20962c25cb4c"),
  name: 'Aurora',
  loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

b. db.unicorns.find({gender:"f", loves:"carrot"}).limit(1)

```
unicorns> db.unicorns.find({gender:"f", loves:"carrot"}).limit(1)
[
  {
    _id: ObjectId("651c43bf938a20962c25cb4c"),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

Практическое задание 8.1.3:

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

db.unicorns.findOne({gender:"f", loves:"carrot"})

```
unicorns> db.unicorns.findOne({gender:"f", loves:"carrot"})
{
  _id: ObjectId("651c43bf938a20962c25cb4c"),
  name: 'Aurora',
  loves: [ 'carrot', 'grape', 'sugar', 'lemons' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

db.unicorns.find().sort({$natural:-1})

```
unicorns> db.unicorns.find().sort({$natural:-1})
[
  {
    _id: ObjectId("652e560d49deea222ad5d1d5"),
    name: 'Nimue',
    dob: ISODate("1999-12-20T13:15:00.000Z"),
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("652e560c49deea222ad5d1d4"),
    name: 'Pilot',
    dob: ISODate("1997-03-01T02:03:00.000Z"),
    loves: [ 'apple', 'watermelon' ],
    weight: 650,
    gender: 'm',
    vampires: 54
  },
  {
    _id: ObjectId("652e560c49deea222ad5d1d3"),
    name: 'Leia',
    dob: ISODate("2001-10-08T10:53:00.000Z"),
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId("652e560c49deea222ad5d1d2"),
    name: 'Raleigh',
    dob: ISODate("2005-05-02T20:57:00.000Z"),
    loves: [ 'apple', 'sugar' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  },
  {
    _id: ObjectId("652e560c49deea222ad5d1d1"),
    name: 'Kenny',
    dob: ISODate("1997-07-01T06:42:00.000Z"),
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
```

Практическое задание 8.1.5:

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

```
db.unicorns.find({}, {loves : {$slice : 1}, _id:0})
```

```
unicorns> 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',
    loves: [ 'apple' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    name: 'Ayna',
    loves: [ 'strawberry' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
```

Практическое задание 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}})

```
unicorns> db.unicorns.find({gender : "f", weight:{$gte : 500, $lte : 700}})
[
  {
    _id: ObjectId("651c43bf938a20962c25cb4f"),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb53"),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb55"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("652e560c49deea222ad5d1cf"),
    name: 'Solnara',
    dob: ISODate("1985-07-03T22:01:00.000Z"),
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId("652e560c49deea222ad5d1d3"),
    name: 'Leia',
    dob: ISODate("2001-10-08T10:53:00.000Z"),
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
```

Практическое задание 8.1.7:

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

db.unicorns.find({gender:"m", $and : [{weight:{$gte:500}}, {loves: {$all : ["grape", "lemon"]}}]}, {_id:0})

```
unicorns> db.unicorns.find({gender:"m", $and : [{weight:{$gte:500}}, {loves: {$all : ["grape", "lemon"]}}]}, {_id:0})
[
  {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    name: 'Kenny',
    dob: ISODate("1997-07-01T06:42:00.000Z"),
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
]
```

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ `vampires`

`db.unicorns.find({vampires:{$exists:false}})`

```
unicorns> db.unicorns.find({vampires:{$exists:false}})
[
  {
    _id: ObjectId("651c43bf938a20962c25cb55"),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
  {
    _id: ObjectId("652e560d49deea222ad5d1d5"),
    name: 'Nimue',
    dob: ISODate("1999-12-20T13:15:00.000Z"),
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

Практическое задание 8.1.9:

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

`db.unicorns.find({gender:"m"}, {loves:{$slice:1}}).sort({name:1})`

```
unicorns> db.unicorns.find({gender:"m"}, {loves:{$slice:1}}).sort({name:1})
[
  {
    _id: ObjectId("651c43fd938a20962c25cb56"),
    name: 'Dunx',
    loves: [ 'grape' ],
    weight: 704,
    gender: 'm',
    vampires: 165,
    zone: DBRef("Zones", 'gf')
  },
  {
    _id: ObjectId("651c43be938a20962c25cb4b"),
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("652e560b49deea222ad5d1cb"),
    name: 'Horny',
    dob: ISODate("1992-03-13T04:47:00.000Z"),
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId("651c43bf938a20962c25cb51"),
    name: 'Kenny',
    loves: [ 'grape' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
  {
    _id: ObjectId("652e560c49deea222ad5d1d1"),
    name: 'Kenny',
    dob: ISODate("1997-07-01T06:42:00.000Z"),
    loves: [ 'grape' ],
    weight: 690,
    gender: 'm',
    vampires: 39
  },
```

Практическое задание 8.2.1:

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

```
{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"}}
```

2) *Сформировать запрос, который возвращает список городов с независимыми мэрами (*`party="I"`*). 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.*

1. OK
2. db.towns.find({"mayor.party":"I"}, {name:1, mayor:1})

```
towns> db.towns.find({"mayor.party":"I"}, {name:1, mayor:1})
[
  {
    _id: ObjectId("65254014615b52563cbf6478"),
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId("65254685615b52563cbf647c"),
    name: 'New York',
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  }
]
```

3. db.towns.find({"mayor.party":{$exists:false}}, {name:1, mayor:1})

```
towns> db.towns.find({"mayor.party":{$exists:false}}, {name:1, mayor:1})
```

Практическое задание 8.2.2:

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`*:*

1. *fn = function () { var c = db.unicorns.find({ gender: "m" }); while (c.hasNext()) { obj = cursor.next(); print(obj); } }*
2. *var cursor = db.unicorns.find({gender:"m"}).limit(2).sort({name:1})*
3. *cursor.forEach(function(obj){print(obj.name)})*

```
unicorns> cursor.forEach(function(obj){print(obj.name)})
Dunx
Horny
```

4. *OK*

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

db.unicorns.find({gender:"f", weight:{$gte : 500, $lte : 600}}).count()

```
unicorns> db.unicorns.find({gender:"f", weight:{$gte : 500, $lte : 600}}).count()
4
```

Практическое задание 8.2.4:

Вывести список предпочтений.

db.unicorns.distinct("loves")

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

Практическое задание 8.2.5:

Посчитать количество особей единорогов обоих полов.

db.unicorns.aggregate({"$group": { _id : "$gender", count:{$sum:1}}})

```
unicorns> db.unicorns.aggregate({"$group": {_id : "$gender", count:{$sum:1}}})
[ { _id: 'f', count: 10 }, { _id: 'm', count: 13 } ]
```

Практическое задание 8.2.6:

1. *Выполнить команду:*

```
> db.unicorns.save({name: 'Barny', loves: ['grape'],
weight: 340, gender: 'm'})
```

2. *Проверить содержимое коллекции* `unicorns`.

1. OK
2. OK

Практическое задание 8.2.7:

1. *Для самки единорога* `Ayna` *внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.*
2. *Проверить содержимое коллекции* `unicorns`.

db.unicorns.updateOne({gender:"f", name:"Ayna"}, {$set:{weight:800, vampires:51}}, {upsert:true})

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

Практическое задание 8.2.8:

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

1. db.unicorns.update({name:"Raleigh"}, {$set:{loves : ["redbull"]}})

```
unicorns> 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
}
```

2. OK

Практическое задание 8.2.9:

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

db.unicorns.update({geder:"m"}, {$inc:{vampires:5}})

Практическое задание 8.2.10:

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

```
db.towns.update({name:'Portland'}, {$unset : {"mayor.party" : 1}})
```

Практическое задание 8.2.11:

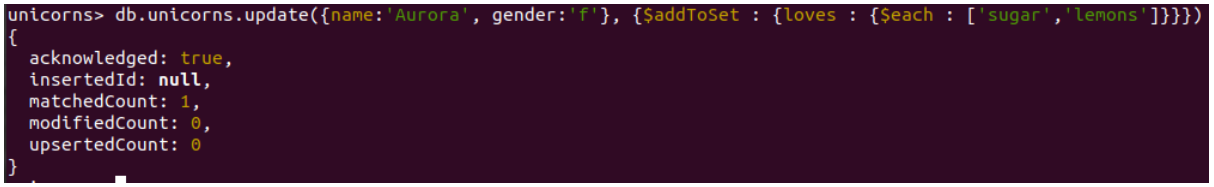
1. *Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.*
2. *Проверить содержимое коллекции* `unicorns`.

```
db.unicorns.update({name:'Pilot',gender:'m'},
{$push:{loves:'chocolate'}})
```

Практическое задание 8.2.12:

1. *Изменить информацию о самке единорога* `Aurora`: *теперь она любит еще и сахар, и лимоны.*
2. *Проверить содержимое коллекции* `unicorns`.

```
db.unicorns.update({name:'Aurora',          gender:'f'},
{$addToSet : {loves : {$each : ['sugar','lemons']}}})
```



Практическое задание 8.2.13:

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

```
{name: "Punxsutawney ",
popujatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: ["phil the groundhog"],
mayor: {
   name: "Jim Wehrle"
   }}

{name: "New York",
popujatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
   name: "Michael Bloomberg",
```

```
   party: "I"}}

{name: "Portland",
popujatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
   name: "Sam Adams",
   party: "D"}}
```

2) *Удалите документы с беспартийными мэрами.*
3) *Проверьте содержание коллекции.*
4) *Очистите коллекцию.*
5) *Просмотрите список доступных коллекций.*

db.towns.deleteMany({"mayor.party" : {$exists : 0}})

```
towns> show collections
towns
```

Практическое задание 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.*
3) *Проверьте содержание коллекции едиорогов.*

db.createCollection("Zones")

db.Zones.insert({_id:"gf", name:"Green Field"})

db.unicorns.update({name:"Dunx"}, {$set:{zone:{$ref:"Zones", $id : "gf"}}})

Практическое задание 8.3.2:

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

db.unicorns.ensureIndex({"name":1}, {"unique":true});

MongoServerError: Index build failed: 64e03e70-815f-4289-920f-f04635ff3e36: Collection unicorns.unicorns (fe494e3b-9c17-4b88-8ffa-7c8c0b26a204) :: caused by :: E11000 duplicate key error collection: unicorns.unicorns index: name_1 dup key: { name: "Aurora" }

```
unicorns> db.unicorns.ensureIndex({"name":1}, {"unique":true});
MongoServerError: Index build failed: 3c4ef372-37fb-4508-a550-d9fc2502ba77: Collection unicorns.unicorns ( fe494e3b-9c17-4b88-8ffa-7c8c0b26a204 ) :: caused by :: E11000 duplicate key error collection: unicorns.unicorns index: name_1 dup key: { name: "Aurora" }
```

Практическое задание 8.3.3:

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

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

3) *Попытайтесь удалить индекс для идентификатора.*

db.unicorns.getIndexes()

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

db.unicorns.dropIndexes()

```
unicorns> db.unicorns.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
unicorns> db.unicorns.dropIndexes()
{
  nIndexesWas: 1,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
```

Практическое задание 8.3.4:

1) *Создайте объемную коллекцию* `numbers`*, задействовав курсор:*

```
for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
```

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* `numbres`*.*

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?*

1. *OK*
2. *db.numbers.find().skip(99996)*

```
numbers> db.numbers.find().skip(99996)
[
  { _id: ObjectId("652e5edf49deea222ad76d52"), value: 99996 },
  { _id: ObjectId("652e5edf49deea222ad76d53"), value: 99997 },
  { _id: ObjectId("652e5edf49deea222ad76d54"), value: 99998 },
  { _id: ObjectId("652e5edf49deea222ad76d55"), value: 99999 }
]
```

3. *db.numbers.explain("executionStats").find({"value" : {$gte : 99996}}).skip(99996)*

```
numbers> db.numbers.explain("executionStats").find({"value" : {$gte : 99996}}).skip(99996)
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'numbers.numbers',
    indexFilterSet: false,
    parsedQuery: { value: { '$gte': 99996 } },
    queryHash: '585F7EDB',
    planCacheKey: 'CFB90839',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      queryPlan: {
        stage: 'FETCH',
        planNodeId: 3,
        inputStage: {
          stage: 'SKIP',
          planNodeId: 2,
          skipAmount: 99996,
          inputStage: {
            stage: 'IXSCAN',
            planNodeId: 1,
            keyPattern: { value: 1 },
            indexName: 'value_1',
            isMultiKey: false,
            multiKeyPaths: { value: [] },
            isUnique: false,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'forward',
            indexBounds: { value: [ '[99996, inf.0]' ] }
          }
        }
      },
      slotBasedPlan: {
        slots: '$$RESULT=s11 env: { s3 = 1697548329678 (NOW), s6 = KS(33FFFFFFFFFFFFFFFE04), s5 = KS(2D03
a/Knox...America/Argentina/Cordoba) (timeZoneDB) }',
        stages: '[3] nlj inner [] [s4, s7, s8, s9, s10] \n' +
          '    left \n' +
          '        [2] limitskip none 99996 \n' +
```

executionTimeMillis = 29

4. *db.numbers.ensureIndex({"value":1})*

```
numbers> db.numbers.ensureIndex({"value":1})
[ 'value_1' ]
```

5. *db.numbers.getIndexes()*

```
numbers> db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
```

6. *OK*
7. *db.numbers.explain("executionStats").find({"value" : {$gte : 99996}}).skip(99996)*

```
numbers> db.numbers.explain("executionStats").find({"value" : {$gte : 99996}}).skip(99996)
{
  explainVersion: '2',
  queryPlanner: {
    namespace: 'numbers.numbers',
    indexFilterSet: false,
    parsedQuery: { value: { '$gte': 99996 } },
    queryHash: '585F7ED8',
    planCacheKey: 'CFB90839',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      queryPlan: {
        stage: 'FETCH',
        planNodeId: 3,
        inputStage: {
          stage: 'SKIP',
          planNodeId: 2,
          skipAmount: 99996,
          inputStage: {
            stage: 'IXSCAN',
            planNodeId: 1,
            keyPattern: { value: 1 },
            indexName: 'value_1',
            isMultiKey: false,
            multiKeyPaths: { value: [] },
            isUnique: false,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'forward',
            indexBounds: { value: [ '[99996, inf.0]' ] }
          }
        }
      },
      slotBasedPlan: {
        slots: '$$RESULT=s11 env: { s3 = 1697548444355 (NOW), s6 = KS(33FFFFFFFFFFFFFFFE04), s5 = KS(2D030D38
a/Knox...America/Argentina/Cordoba) (timeZoneDB) }',
        stages: '[3] nlj inner [] [s4, s7, s8, s9, s10] \n' +
          '    left \n' +
          '        [2] limitskip none 99996 \n' +
          '        [1] cfilter {(exists(s5) && exists(s6))} \n' +
          '        [1] ixseek s5 s6 s9 s4 s7 s8 [] @"55e5ed7d-d1a1-4581-b90b-745f236f3ef1" @"value_1" true \n' +
          '    right \n' +
```

executionTimeMillis = 10

8. *На таком большом массиве данных при поиске по заданному ключу value более эффективным оказался способ с индексацией. Разница ощутима : почти в 3 раза*