

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 8

**по теме: Запросы на выборку данных к БД PostgreSQL.
Представления в PostgreSQL**

по дисциплине: Проектирование и реализация баз данных

Специальность:

45.03.04 Интеллектуальные системы в гуманитарной сфере

Проверил:

Говорова М.М. _____

Дата: «16» марта 2021г.

Оценка _____

Выполнила:

студентка группы К3243

Дорофеева Арина

Санкт-Петербург 2020/2021

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.

Ход работы

Практическое задание 8.1.1:

1) Создайте базу данных learn.

2) Заполните коллекцию единорогов 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'});
```

3) Используя второй способ, вставьте в коллекцию единорогов документ:

```
{name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
```

```
> 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)
WriteResult({ "nInserted" : 1 })
> db.unicorns.find({"name": "Dunx"})
{ "id" : ObjectId("60bcc92350056ae513106be9"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
{ "id" : ObjectId("60bcc9c850056ae513106bea"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
> █
```

4) Проверьте содержимое коллекции с помощью метода find.

```
> db.unicorns.find()
{ "_id" : ObjectId("60bcc83750056ae513106bdd"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" : 63 }
{ "_id" : ObjectId("60bcc83850056ae513106bde"), "name" : "Aurora", "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" : "f", "vampires" : 43 }
{ "_id" : ObjectId("60bcc83850056ae513106bdf"), "name" : "Unicrom", "loves" : [ "energion", "redbull" ], "weight" : 984, "gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("60bcc83850056ae513106be0"), "name" : "Rooooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 99 }
{ "_id" : ObjectId("60bcc83850056ae513106be1"), "name" : "Solnara", "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 550, "gender" : "f", "vampires" : 80 }
{ "_id" : ObjectId("60bcc83850056ae513106be2"), "name" : "Ayna", "loves" : [ "strawberry", "lemon" ], "weight" : 733, "gender" : "f", "vampires" : 40 }
{ "_id" : ObjectId("60bcc83850056ae513106be3"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "_id" : ObjectId("60bcc83850056ae513106be4"), "name" : "Raleigh", "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "_id" : ObjectId("60bcc83850056ae513106be5"), "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender" : "f", "vampires" : 33 }
{ "_id" : ObjectId("60bcc83850056ae513106be6"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("60bcc83850056ae513106be7"), "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
{ "_id" : ObjectId("60bcc92350056ae513106be9"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
{ "_id" : ObjectId("60bcc9c850056ae513106bea"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
> █
```

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.

```
> db.unicorns.find({"gender": "m"}, {"name": 1, "_id": 0}).sort({"name": 1})
{ "name" : "Dunx" }
{ "name" : "Dunx" }
{ "name" : "Horny" }
{ "name" : "Kenny" }
{ "name" : "Pilot" }
{ "name" : "Raleigh" }
{ "name" : "Rooooooodles" }
{ "name" : "Unicrom" }
> █
> db.unicorns.find({"gender": "f"}, {"name": 1, "_id": 0}).sort({"name": 1}).limit(3)
{ "name" : "Aurora" }
{ "name" : "Ayna" }
{ "name" : "Leia" }
> █
```

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"}, {"name": 1, "_id": 0})
{ "name" : "Aurora" }
> █
> db.unicorns.find({"gender": "f", "loves": "carrot"}, {"name": 1, "_id": 0}).limit(1)
{ "name" : "Aurora" }
> █
```

Практическое задание 8.1.3:

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

```
> db.unicorns.find({"gender": "m"}, {"loves": 0})
{ "_id" : ObjectId("60bcc83750056ae513106bdd"), "name" : "Horny", "weight" : 600, "gender" : "m", "vampires" : 63 }
{ "_id" : ObjectId("60bcc83850056ae513106bdf"), "name" : "Unicrom", "weight" : 984, "gender" : "m", "vampires" : 182 }
{ "_id" : ObjectId("60bcc83850056ae513106be0"), "name" : "Rooooooodles", "weight" : 575, "gender" : "m", "vampires" : 99 }
{ "_id" : ObjectId("60bcc83850056ae513106be3"), "name" : "Kenny", "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "_id" : ObjectId("60bcc83850056ae513106be4"), "name" : "Raleigh", "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "_id" : ObjectId("60bcc83850056ae513106be6"), "name" : "Pilot", "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("60bcc9c850056ae513106beb"), "name" : "Dunx", "weight" : 704, "gender" : "m", "vampires" : 165 }
> █
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find({}, {"name": 1, "_id": 0}).sort({$natural: -1})
{ "name" : "Dunx" }
{ "name" : "Nimue" }
{ "name" : "Pilot" }
{ "name" : "Leia" }
{ "name" : "Raleigh" }
{ "name" : "Kenny" }
{ "name" : "Ayna" }
{ "name" : "Solnara" }
{ "name" : "Roooooodles" }
{ "name" : "Unicrom" }
{ "name" : "Aurora" }
{ "name" : "Horny" }
> █
```

Практическое задание 8.1.5:

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 }
{ "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 }
{ "name" : "Kenny", "loves" : [ "grape" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "name" : "Raleigh", "loves" : [ "apple" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "name" : "Leia", "loves" : [ "apple" ], "weight" : 601, "gender" : "f", "vampires" : 33 }
{ "name" : "Pilot", "loves" : [ "apple" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "name" : "Nimue", "loves" : [ "grape" ], "weight" : 540, "gender" : "f" }
{ "name" : "Dunx", "loves" : [ "grape" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
> █
```

Практическое задание 8.1.6:

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

```
> db.unicorns.find({"weight": {$gte: 500, $lte: 700}}, {"_id": 0})
{ "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" : 63 }
{ "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 99 }
{ "name" : "Solnara", "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 550, "gender" : "f", "vampires" : 80 }
{ "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender" : "f", "vampires" : 33 }
{ "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
> █
```

Практическое задание 8.1.7:

Вывести список самцов единорогов весом от полутонны и предпочитающих грейп и лимон, исключив вывод идентификатора.

```
> db.unicorns.find({"gender": "m", "weight": {$gte: 500}, $or: [{"loves": "grape"}, {"loves": "lemon"}]}, {"_id": 0})
{ "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
> █
```

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ `vampires`.

```
> db.unicorns.find({"vampires": {$exists: false}}, {"_id": 0})
{ "name" : "Nimue", "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"}, {"_id": 0, "name": 1, "loves": 1}).sort({"name": 1})
{ "name" : "Dunx", "loves" : [ "grape", "watermelon" ] }
{ "name" : "Horny", "loves" : [ "carrot", "papaya" ] }
{ "name" : "Kenny", "loves" : [ "grape", "lemon" ] }
{ "name" : "Pilot", "loves" : [ "apple", "watermelon" ] }
{ "name" : "Raleigh", "loves" : [ "apple", "sugar" ] }
{ "name" : "Rooodoodles", "loves" : [ "apple" ] }
{ "name" : "Unicrom", "loves" : [ "energon", "redbull" ] }
> █
```

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.

```

6 db.towns.insertMany([{name: "Punxsutawney ",
7 population: 6200,
8 last_sensus: ISODate("2008-01-31"),
9 famous_for: [""],
10 mayor: {
11 |   name: "Jim Wehrle"
12 }},
13
14 {name: "New York",
15 population: 22200000,
16 last_sensus: ISODate("2009-07-31"),
17 famous_for: ["status of liberty", "food"],
18 mayor: {
19 |   name: "Michael Bloomberg",
20 party: "I"}},
21
22 {name: "Portland",
23 population: 528000,
24 last_sensus: ISODate("2009-07-20"),
25 famous_for: ["beer", "food"],
26 mayor: {
27 |   name: "Sam Adams",
28 party: "D"}}})
29

```

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

```

> db.towns.find({"mayor.party": "I"}, {"_id": 0, "name": 1, "mayor": 1})
{ "name" : "New York", "mayor" : { "name" : "Michael Bloomberg", "party" : "I" } }
> █

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

```

Практическое задание 8.2.2:

- 1) Сформировать функцию для вывода списка самцов единорогов.

```

> fn = function() {return this.gender=="m"; }
function() {return this.gender=="m"; }
> db.unicorns.find(fn)
{ "_id" : ObjectId("60bcc83750056ae513106bdd"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" : 63 }
{ "_id" : ObjectId("60bcc83850056ae513106bdf"), "name" : "Umicrom", "loves" : [ "energon", "redbull" ], "weight" : 984, "gender" : "m", "vampire s" : 192 }
{ "_id" : ObjectId("60bcc83850056ae513106be0"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 99 }
{ "_id" : ObjectId("60bcc83850056ae513106be3"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 39 }
{ "_id" : ObjectId("60bcc83850056ae513106be4"), "name" : "Raleigh", "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "_id" : ObjectId("60bcc83850056ae513106be6"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
{ "_id" : ObjectId("60bccfea50056ae513106beb"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires" : 165 }
> █

```

- 2) Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.
- 3) Вывести результат, используя forEach.

```

> fn = function() {return this.gender=="m"; }
function() {return this.gender=="m"; }
> var cursor = db.unicorns.find(fn).limit(2).sort({"name": 1})
> cursor.forEach((unicorn) => {print(unicorn.name)})
Dunx
Horny
> █

```

4) Содержание коллекции единорогов 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', 44), 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'});

db.unicorns.insert ({name: 'Dunx', loves: ['grape', 'watermelon'],
weight: 704, gender: 'm', vampires: 165})
```

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
> db.unicorns.find({"gender": "f", "weight": {$lte: 600}}).count()
3
> █
```

Практическое задание 8.2.4:

Вывести список предпочтений.

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

Практическое задание 8.2.5:

Посчитать количество особей единорогов обоих полов

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

Практическое задание 8.2.6:

1. Выполнить команду:

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

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.find({name: "Barney"})
> db.unicorns.save({name: 'Barney', loves: ['grape'], weight: 340, gender: 'm'})
WriteResult({ "nInserted" : 1 })
> db.unicorns.find({name: "Barney"})
{ "_id" : ObjectId("60bcef20d2c12a9f8359fe39"), "name" : "Barney", "loves" : [ "grape" ], "weight" : 340, "gender" : "m" }
> █
```

Практическое задание 8.2.7:

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

```
> db.unicorns.find({name: "Ayna"})
{ "_id" : ObjectId("60bcf090d2c12a9f8359fe3a"), "name" : "Ayna", "loves" : [ "strawberry", "lemon" ], "weight" : 733, "gender" : "f", "vampires" : 40 }
> db.unicorns.update({name: "Ayna"}, {name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 800, gender: 'f', vampires: 51})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.find({name: "Ayna"})
{ "_id" : ObjectId("60bcf090d2c12a9f8359fe3a"), "name" : "Ayna", "loves" : [ "strawberry", "lemon" ], "weight" : 800, "gender" : "f", "vampires" : 51 }
> █
```

Практическое задание 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", gender: "m"})
{ "_id" : ObjectId("60bcc83850056ae513106be4"), "name" : "Raleigh", "loves" : [ "apple", "sugar" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
> db.unicorns.update({name: "Raleigh", gender: "m"}, {$set: {loves: ["redbull"]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.find({name: "Raleigh", gender: "m"})
{ "_id" : ObjectId("60bcc83850056ae513106be4"), "name" : "Raleigh", "loves" : [ "redbull" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
> █
```

Практическое задание 8.2.9:

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


```

> db.unicorns.find({gender: "m"}, {vampires: 1})
{ "_id" : ObjectId("60bcc83750056ae513106bdd"), "vampires" : 63 }
{ "_id" : ObjectId("60bcc83850056ae513106bdf"), "vampires" : 182 }
{ "_id" : ObjectId("60bcc83850056ae513106be0"), "vampires" : 99 }
{ "_id" : ObjectId("60bcc83850056ae513106be3"), "vampires" : 39 }
{ "_id" : ObjectId("60bcc83850056ae513106be4"), "vampires" : 2 }
{ "_id" : ObjectId("60bcc83850056ae513106be6"), "vampires" : 54 }
{ "_id" : ObjectId("60bccfea50056ae513106beb"), "vampires" : 165 }
{ "_id" : ObjectId("60bcef20d2c12a9f8359fe39") }
> db.unicorns.update({gender: "m"}, {$inc: {vampires: 5}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.find({gender: "m"}, {vampires: 1})
{ "_id" : ObjectId("60bcc83750056ae513106bdd"), "vampires" : 68 }
{ "_id" : ObjectId("60bcc83850056ae513106bdf"), "vampires" : 182 }
{ "_id" : ObjectId("60bcc83850056ae513106be0"), "vampires" : 99 }
{ "_id" : ObjectId("60bcc83850056ae513106be3"), "vampires" : 39 }
{ "_id" : ObjectId("60bcc83850056ae513106be4"), "vampires" : 2 }
{ "_id" : ObjectId("60bcc83850056ae513106be6"), "vampires" : 54 }
{ "_id" : ObjectId("60bccfea50056ae513106beb"), "vampires" : 165 }
{ "_id" : ObjectId("60bcef20d2c12a9f8359fe39") }
> █

```

Практическое задание 8.2.10:

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

```

> db.towns.find({name: "Portland"})
{ "_id" : ObjectId("60bcd88db81d6243c653a4c"), "name" : "Portland", "population" : 528000, "last_sensus" : ISODate("2009-07-20T00:00:00Z"), "famous_for" : [ "beer", "food" ], "mayor" : { "name" : "Sam Adams", "party" : "D" } }
> db.towns.update({name: "Portland"}, {$unset: {"mayor.party": 1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.towns.find({name: "Portland"})
{ "_id" : ObjectId("60bcd88db81d6243c653a4c"), "name" : "Portland", "population" : 528000, "last_sensus" : ISODate("2009-07-20T00:00:00Z"), "famous_for" : [ "beer", "food" ], "mayor" : { "name" : "Sam Adams" } }
> █

```

Практическое задание 8.2.11:

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

```

> db.unicorns.find({name: "Pilot"})
{ "_id" : ObjectId("60bcc83850056ae513106be6"), "name" : "Pilot", "loves" : [ "apple", "watermelon" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
> db.unicorns.update({name: "Pilot"}, {$push: {loves: "chocolate"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.find({name: "Pilot"})
{ "_id" : ObjectId("60bcc83850056ae513106be6"), "name" : "Pilot", "loves" : [ "apple", "watermelon", "chocolate" ], "weight" : 650, "gender" : "m", "vampires" : 54 }
> █

```

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.find({name: "Aurora"})
{ "_id" : ObjectId("60bcc83850056ae513106bde"), "name" : "Aurora", "loves" : [ "carrot", "grape" ], "weight" : 450, "gender" : "f", "vampires" : 43 }
> db.unicorns.update({name: "Aurora"}, {$addToSet: {"loves": {$each: ["sugar", "lemon"]}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.find({name: "Aurora"})
{ "_id" : ObjectId("60bcc83850056ae513106bde"), "name" : "Aurora", "loves" : [ "carrot", "grape", "sugar", "lemon" ], "weight" : 450, "gender" : "f", "vampires" : 43 }
> █

```

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.

```
> db.towns.find()
{ "_id" : ObjectId("60bcfa43fce444937f9b28fe"), "name" : "Punxsutawney", "population" : 6200, "last_sensus" : ISODate("2008-01-31T00:00:00Z"), "famous_for" : [ "phil the groundhog" ], "mayor" : { "name" : "Jim Wehrle" } }
{ "_id" : ObjectId("60bcfa43fce444937f9b28ff"), "name" : "New York", "population" : 22200000, "last_sensus" : ISODate("2009-07-31T00:00:00Z"), "famous_for" : [ "statue of liberty", "food" ], "mayor" : { "name" : "Michael Bloomberg", "party" : "I" } }
{ "_id" : ObjectId("60bcfa43fce444937f9b2900"), "name" : "Portland", "population" : 528000, "last_sensus" : ISODate("2009-07-20T00:00:00Z"), "famous_for" : [ "beer", "food" ], "mayor" : { "name" : "Sam Adams", "party" : "D" } }
> db.towns.remove({"mayor.party": {$exists: false}})
WriteResult({ "nRemoved" : 1 })
> db.towns.find()
{ "_id" : ObjectId("60bcfa43fce444937f9b28ff"), "name" : "New York", "population" : 22200000, "last_sensus" : ISODate("2009-07-31T00:00:00Z"), "famous_for" : [ "statue of liberty", "food" ], "mayor" : { "name" : "Michael Bloomberg", "party" : "I" } }
{ "_id" : ObjectId("60bcfa43fce444937f9b2900"), "name" : "Portland", "population" : 528000, "last_sensus" : ISODate("2009-07-20T00:00:00Z"), "famous_for" : [ "beer", "food" ], "mayor" : { "name" : "Sam Adams", "party" : "D" } }
> █
```

4) Очистите коллекцию.

5) Просмотрите список доступных коллекций.

```
> db.towns.drop()
true
> show collections
unicorns
> █
```

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 единорогов.

```

> db.zones.insertMany([
...   {id: "forest",
...     name: "forest",
...     description: "A forest with lots of trees"},
...   {id: "b_forest",
...     name: "birch forest",
...     description: "A forest with lots of birch trees"},
...   {id: "j_forest",
...     name: "jungle",
...     description: "A forest with lots of jungle trees"},
...   {id: "t_forest",
...     name: "taiga",
...     description: "A forest with lots of spruce trees"}
... ])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("60bd2289fce444937f9b2905"),
    ObjectId("60bd2289fce444937f9b2906"),
    ObjectId("60bd2289fce444937f9b2907"),
    ObjectId("60bd2289fce444937f9b2908")
  ]
}
> db.zones.find()
{ "_id" : ObjectId("60bd2289fce444937f9b2905"), "id" : "forest", "name" : "forest", "description" : "A forest with lots of trees" }
{ "_id" : ObjectId("60bd2289fce444937f9b2906"), "id" : "b_forest", "name" : "birch forest", "description" : "A forest with lots of birch trees" }
{ "_id" : ObjectId("60bd2289fce444937f9b2907"), "id" : "j_forest", "name" : "jungle", "description" : "A forest with lots of jungle trees" }
{ "_id" : ObjectId("60bd2289fce444937f9b2908"), "id" : "t_forest", "name" : "taiga", "description" : "A forest with lots of spruce trees" }
> █

> db.unicorns.update({"name": "Leia"}, {$set: {zone: {$ref: "zones", $id: "j_forest"}}})
WriteResult({"nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.unicorns.find({"name": "Leia"})
{ "_id" : ObjectId("60bcc83850056ae513106be5"), "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender" : "f", "vampires" : 33, "zone" : DBRef("zones", "j_forest") }
> █

```

4) Содержание коллекции единорогов 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', 44), 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'});
```

```
db.unicorns.insert {name: 'Dunx', loves: ['grape', 'watermelon'], weight:
704, gender: 'm', vampires: 165}
```

```
> db.unicorns.find()
{ "_id" : ObjectId("60bcc83750056ae513106bdd"), "name" : "Horny", "loves" : [ "carrot", "papaya" ], "weight" : 600, "gender" : "m", "vampires" :
68, "zone" : DBRef("zones", "forest") }
{ "_id" : ObjectId("60bcc83850056ae513106bde"), "name" : "Aurora", "loves" : [ "carrot", "grape", "sugar", "lemon" ], "weight" : 450, "gender" :
"f", "vampires" : 43, "zone" : DBRef("zones", "j_forest") }
{ "_id" : ObjectId("60bcc83850056ae513106bdf"), "name" : "Unicrom", "loves" : [ "energon", "redbull" ], "weight" : 984, "gender" : "m", "vampire
s" : 182 }
{ "_id" : ObjectId("60bcc83850056ae513106be0"), "name" : "Roooooodles", "loves" : [ "apple" ], "weight" : 575, "gender" : "m", "vampires" : 99 }
{ "_id" : ObjectId("60bcc83850056ae513106be1"), "name" : "Solnara", "loves" : [ "apple", "carrot", "chocolate" ], "weight" : 550, "gender" : "f",
"vampires" : 80 }
{ "_id" : ObjectId("60bcc83850056ae513106be3"), "name" : "Kenny", "loves" : [ "grape", "lemon" ], "weight" : 690, "gender" : "m", "vampires" : 3
9 }
{ "_id" : ObjectId("60bcc83850056ae513106be4"), "name" : "Raleigh", "loves" : [ "redbull" ], "weight" : 421, "gender" : "m", "vampires" : 2 }
{ "_id" : ObjectId("60bcc83850056ae513106be5"), "name" : "Leia", "loves" : [ "apple", "watermelon" ], "weight" : 601, "gender" : "f", "vampires"
: 33 }
{ "_id" : ObjectId("60bcc83850056ae513106be6"), "name" : "Pilot", "loves" : [ "apple", "watermelon", "chocolate" ], "weight" : 650, "gender" : "
m", "vampires" : 54, "zone" : DBRef("zones", "j_forest") }
{ "_id" : ObjectId("60bcc83850056ae513106be7"), "name" : "Nimue", "loves" : [ "grape", "carrot" ], "weight" : 540, "gender" : "f" }
{ "_id" : ObjectId("60bcc83850056ae513106beb"), "name" : "Dunx", "loves" : [ "grape", "watermelon" ], "weight" : 704, "gender" : "m", "vampires"
: 165, "zone" : DBRef("zones", "b_forest") }
{ "_id" : ObjectId("60bcef20d2c12a9f8359fe39"), "name" : "Barny", "loves" : [ "grape" ], "weight" : 340, "gender" : "m" }
{ "_id" : ObjectId("60bcf090d2c12a9f8359fe3a"), "name" : "Ayna", "loves" : [ "strawberry", "lemon" ], "weight" : 800, "gender" : "f", "vampires"
: 51, "zone" : DBRef("zones", "b_forest") }
> █
```

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})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
> db.unicorns.insert({name: "Pilot"})
WriteResult({
  "nInserted" : 0,
  "writeError" : {
    "code" : 11000,
    "errmsg" : "E11000 duplicate key error collection: learn.unicorns index: name_1 dup key: { name: \"Pilot\" }"
  }
})
> █
```

2. Содержание коллекции единорогов unicorns:

```
db.unicorns.insert({name: 'Horny', dob: new Date(1992,2,13,7,47), loves:
['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
```

```
db.unicorns.insert({name: 'Aurora', dob: new Date(1991, 0, 24, 13, 0),
loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
```

```
db.unicorns.insert({name: 'Unicrom', dob: new Date(1973, 1, 9, 22, 10),
loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
```

```
db.unicorns.insert({name: 'Roooooodles', dob: new Date(1979, 7, 18, 18,
44), loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
```

```
db.unicorns.insert({name: 'Solnara', dob: new Date(1985, 6, 4, 2, 1),
loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80});
```

```
db.unicorns.insert({name: 'Ayna', dob: new Date(1998, 2, 7, 8, 30), loves:
['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
```

```
db.unicorns.insert({name: 'Kenny', dob: new Date(1997, 6, 1, 10, 42),
loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});
```

```
db.unicorns.insert({name: 'Raleigh', dob: new Date(2005, 4, 3, 0, 57),
loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});
```

```
db.unicorns.insert({name: 'Leia', dob: new Date(2001, 9, 8, 14, 53),
loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});
```

```
db.unicorns.insert({name: 'Pilot', dob: new Date(1997, 2, 1, 5, 3), loves:
['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});
```

```
db.unicorns.insert ({name: 'Nimue', dob: new Date(1999, 11, 20, 16, 15),
loves: ['grape', 'carrot'], weight: 540, gender: 'f'});
```

```
db.unicorns.insert {name: 'Dunx', dob: new Date(1976, 6, 18, 18, 18),
loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 16
```

Практическое задание 8.3.3:

- 1) Получите информацию обо всех индексах коллекции unicorns.

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

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

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

- 3) Попробуйте удалить индекс для идентификатора.

```
> db.unicorns.dropIndex("_id_")
{
  "ok" : 0,
  "errmsg" : "cannot drop _id index",
  "code" : 72,
  "codeName" : "InvalidOptions"
}
```

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.

```

> db.numbers.explain("executionStats").find().sort({value: -1}).limit(4)
{
  "queryPlanner" : {
    "plannerVersion" : 1,
    "namespace" : "learn.numbers",
    "indexFilterSet" : false,
    "parsedQuery" : {

    },
    "winningPlan" : {
      "stage" : "SORT",
      "sortPattern" : {
        "value" : -1
      },
      "memLimit" : 104857600,
      "limitAmount" : 4,
      "type" : "simple",
      "inputStage" : {
        "stage" : "COLLSCAN",
        "direction" : "forward"
      }
    },
    "rejectedPlans" : [ ]
  },
  "executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 4,
    "executionTimeMillis" : 76,
    "totalKeysExamined" : 0,
    "totalDocsExamined" : 100000,
    "executionStages" : {
      "stage" : "SORT",
      "nReturned" : 4,
      "executionTimeMillisEstimate" : 3,

```

- 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`.

```

> db.numbers.ensureIndex({value: 1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 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.explain("executionStats").find().sort({value: -1}).limit(4)
{
  "queryPlanner" : {
    "plannerVersion" : 1,
    "namespace" : "learn.numbers",
    "indexFilterSet" : false,
    "parsedQuery" : {

    },
    "winningPlan" : {
      "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" : 0,
    "totalKeysExamined" : 4,
    "totalDocsExamined" : 4,
    "executionStages" : {
      "stage" : "LIMIT",
      "nReturned" : 4,
      "executionTimeMillisEstimate" : 0,

```

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

После создания индекса время исполнения команды стало равно нулю.

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

Индексирование сократило время выполнения с 76 мс до 0 мс, следовательно, запрос с индексом более эффективен.

Вывод

MongoDB предоставляет мощный CLI интерфейс для выполнения CRUD операций, отличительной особенностью является интеграция полноценного языка программирования: javascript.