ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет «Инфокоммуникационных технологий»
Направление подготовки «45.03.04 Интеллектуальные системы в гуманитарной среде»

О Т Ч Е Т

ЛАБОРАТОРНАЯ РАБОТА №6

Тема задания: Знакомство с MongoDB

Выполнил:
Студент **Янов Ф.А.** **К3243**
(Фамилия И.О.) номер группы

Проверил:
Преподаватель **Говоров А.И.**
(Фамилия И.О)

Санкт-Петербург
2020

Индивидуальное задание:

Создать программную систему, предназначенную для администрации аэропорта некоторой компании-авиаперевозчика.

Рейсы обслуживаются бортами, принадлежащими разным авиаперевозчикам. О каждом самолете необходима следующая минимальная информация: номер самолета, тип, число мест, скорость полета, компания-авиаперевозчик. Один тип самолета может летать на разных маршрутах и по одному маршруту могут летать разные типы самолетов.

О каждом рейсе необходима следующая информация: номер рейса, расстояние до пункта назначения, пункт вылета, пункт назначения; дата и время вылета, дата и время прилета, транзитные посадки (если есть), пункты посадки, дата и время транзитных посадок и дат и время их вылета, количество проданных билетов. Каждый рейс обслуживается определенным экипажем, в состав которого входят командир корабля, второй пилот, штурман и стюардессы или стюарды. Каждый экипаж может обслуживать разные рейсы на разных самолетах. Необходимо предусмотреть наличие информации о допуске члена экипажа к рейсу.

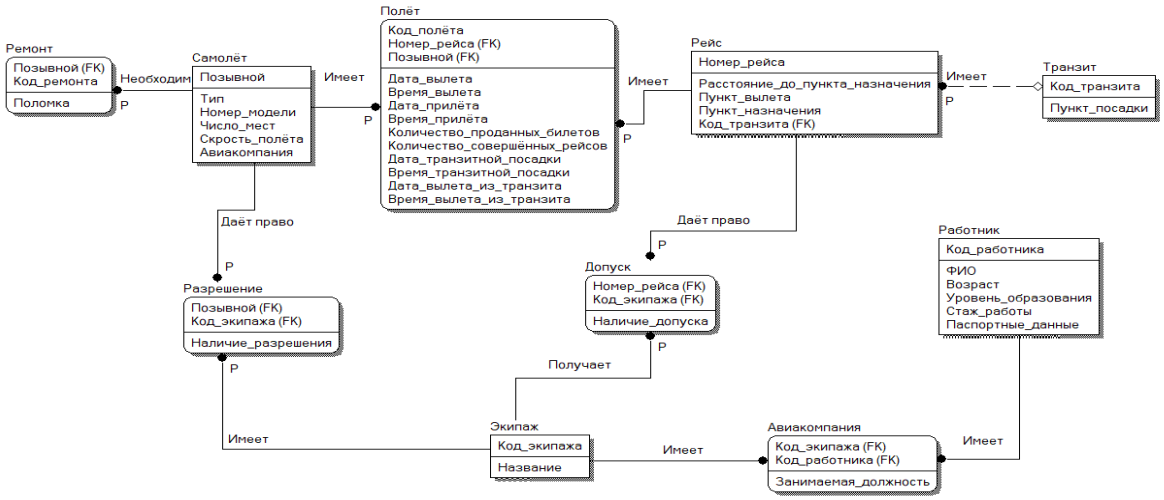
Администрация компании-владельца аэропорта должна иметь возможность принять работника на работу или уволить. При этом необходима следующая информация: ФИО, возраст, образование, стаж работы, паспортные данные. Эта же информация необходима для сотрудников сторонних компаний.

Выполнение работы:

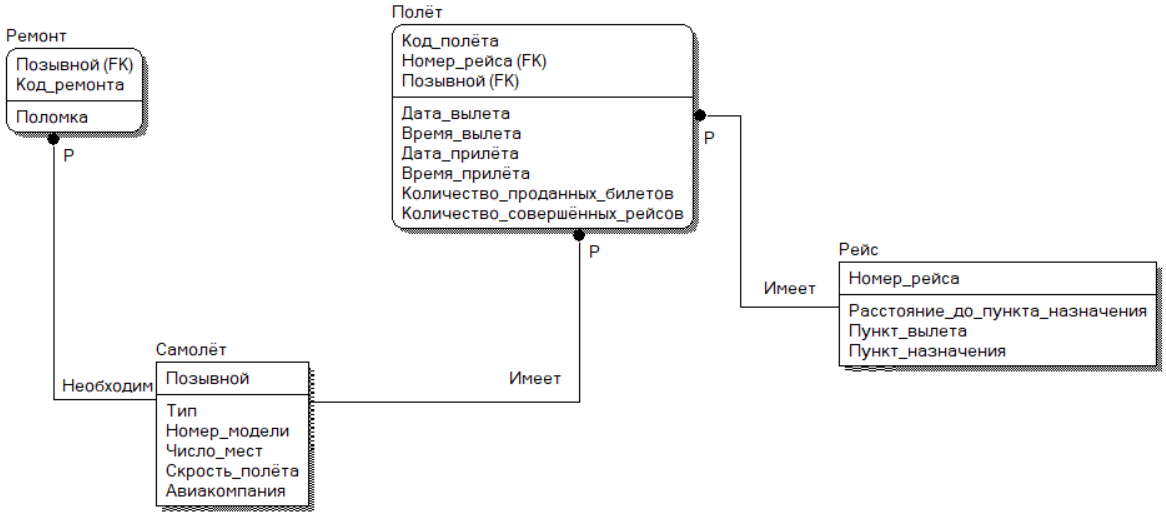
Для выполнения работы Вам потребуется:

1) Попытаться уменьшить размер модели реляционной базы данных, с которой Вы работали.
2) Реализовать Вашу модель в MongoDB.
3) Заполнить модель данными, настолько, чтобы все запросы реализуемые Вами в пункте 2.d выдавали непустые строки.
4) Реализовать минимум 40% Ваших запросов, сделанных в работе по реляционным базам данных в MongoDB.

У нас была база данных во втором задании:



Сейчас нам достаточно выполнить 40% запросов к ней, соответственно, можно просто убрать не нужные сущности. Уменьшим нашу бд согласно требованиям первого пункта:



Как видим, у нас остались только 4 сущности (2 из них были сокращены), но практика показывала, что в 4ой лабораторной, где нужно было выполнить запросы, около половины (что > 40%) было проведено именно с этими таблицами.

Реализуем нашу модель в **MongoDB**;

Создадим таблицу «самолёт» и наполним её данными:

```
> use airport
switched to db airport
> db.samolet.insert({"tip": "Boeing", "nomer_modeli": 738,"chislo_mest": 190, "skorost_poleta": 850, "aviacompany": "S7"})
WriteResult({ "nInserted" : 1 })
> db.samolet.insert({"tip": "Airbus", "nomer_modeli": 320,"chislo_mest": 146, "skorost_poleta": 830, "aviacompany": "Aeroflot"})
WriteResult({ "nInserted" : 1 })
> db.samolet.insert({"tip": "Boeing", "nomer_modeli": 773,"chislo_mest": 550, "skorost_poleta": 950, "aviacompany": "British_airways"})
WriteResult({ "nInserted" : 1 })
> db.samolet.insert({"tip": "Airbus", "nomer_modeli": 359,"chislo_mest": 314, "skorost_poleta": 945, "aviacompany": "Lufthansa"})
WriteResult({ "nInserted" : 1 })
> db.samolet.insert({"tip": "Boeing", "nomer_modeli": 737,"chislo_mest": 178, "skorost_poleta": 840, "aviacompany": "Aeroflot"})
WriteResult({ "nInserted" : 1 })
> db.samolet.find()
{ "_id" : ObjectId("5eff4b7ad82b8666975964d3"), "tip" : "Boeing", "nomer_modeli" : 738, "chislo_mest" : 190, "skorost_poleta" : 850, "aviacompany" : "S7" }
{ "_id" : ObjectId("5eff4f65d82b8666975964d4"), "tip" : "Airbus", "nomer_modeli" : 320, "chislo_mest" : 146, "skorost_poleta" : 830, "aviacompany" : "Aeroflot" }
{ "_id" : ObjectId("5eff4f6bd82b8666975964d5"), "tip" : "Boeing", "nomer_modeli" : 773, "chislo_mest" : 550, "skorost_poleta" : 950, "aviacompany" : "British_airways" }
{ "_id" : ObjectId("5eff4f70d82b8666975964d6"), "tip" : "Airbus", "nomer_modeli" : 359, "chislo_mest" : 314, "skorost_poleta" : 945, "aviacompany" : "Lufthansa" }
{ "_id" : ObjectId("5eff4f74d82b8666975964d7"), "tip" : "Boeing", "nomer_modeli" : 737, "chislo_mest" : 178, "skorost_poleta" : 840, "aviacompany" : "Aeroflot" }
>
```

Теперь также заполним таблицу «рейс»:

```
> db.reys.insert({"rasstoyanie_do_punkta_naznachenia": 409, "punkt_vyleta": "Vnukovo","punkt_priletat": "Strigino"})
WriteResult({ "nInserted" : 1 })
> db.reys.insert({"rasstoyanie_do_punkta_naznachenia": 34, "punkt_vyleta": "Domodedovo","punkt_priletat": "Vnukovo"})
WriteResult({ "nInserted" : 1 })
> db.reys.insert({"rasstoyanie_do_punkta_naznachenia": 5330, "punkt_vyleta": "Pulkovo","punkt_priletat": "Lisboa"})
WriteResult({ "nInserted" : 1 })
> db.reys.insert({"rasstoyanie_do_punkta_naznachenia": 305, "punkt_vyleta": "Frankfurt_I_A","punkt_priletat": "Zurich_m_a"})
WriteResult({ "nInserted" : 1 })
> db.reys.find()
{ "_id" : ObjectId("5eff58a5d82b8666975964d8"), "rasstoyanie_do_punkta_naznachenia" : 725, "punkt_vyleta" : "Pulkovo", "punkt_priletat" : "Sheremetievo" }
{ "_id" : ObjectId("5eff58bdd82b8666975964d9"), "rasstoyanie_do_punkta_naznachenia" : 409, "punkt_vyleta" : "Vnukovo", "punkt_priletat" : "Strigino" }
{ "_id" : ObjectId("5eff58c4d82b8666975964da"), "rasstoyanie_do_punkta_naznachenia" : 34, "punkt_vyleta" : "Domodedovo", "punkt_priletat" : "Vnukovo" }
{ "_id" : ObjectId("5eff58c9d82b8666975964db"), "rasstoyanie_do_punkta_naznachenia" : 5330, "punkt_vyleta" : "Pulkovo", "punkt_priletat" : "Lisboa" }
{ "_id" : ObjectId("5eff58cfd82b8666975964dc"), "rasstoyanie_do_punkta_naznachenia" : 305, "punkt_vyleta" : "Frankfurt_I_A", "punkt_priletat" : "Zurich_m_a" }
```

Теперь ремонт:

```
> db.remont.insert({id_samolet: ObjectId("5eff4b7ad82b8666975964d3"), "polomka" : "Nasos"})
WriteResult({ "nInserted" : 1 })
> db.remont.insert({id_samolet: ObjectId("5eff4f65d82b8666975964d4"), "polomka" : "Dvertsa_tualeta"})
WriteResult({ "nInserted" : 1 })
> db.remont.insert({id_samolet: ObjectId("5eff4f6bd82b8666975964d5"), "polomka" : "Zadniya_dver"})
WriteResult({ "nInserted" : 1 })
> db.remont.insert({id_samolet: ObjectId("5eff4f70d82b8666975964d6"), "polomka" : "Chvostovoe_operenie"})
WriteResult({ "nInserted" : 1 })
> db.remont.insert({id_samolet: ObjectId("5eff4f70d82b8666975964d7"), "polomka" : "Shassi"})
WriteResult({ "nInserted" : 1 })
```

```
> db.remont.find()
{ "_id" : ObjectId("5eff714a7d954ddb96164ba6"), "id_samolet" : ObjectId("5eff4b7ad82b8666975964d3"), "polomka" : "Nasos" }
{ "_id" : ObjectId("5eff71517d954ddb96164ba7"), "id_samolet" : ObjectId("5eff4f65d82b8666975964d4"), "polomka" : "Dvertsa_tualeta" }
{ "_id" : ObjectId("5eff71567d954ddb96164ba8"), "id_samolet" : ObjectId("5eff4f6bd82b8666975964d5"), "polomka" : "Zadniya_dver" }
{ "_id" : ObjectId("5eff715d7d954ddb96164ba9"), "id_samolet" : ObjectId("5eff4f70d82b8666975964d6"), "polomka" : "Chvostovoe_operenie" }
{ "_id" : ObjectId("5eff77f67d954ddb96164bac"), "id_samolet" : ObjectId("5eff4f70d82b8666975964d7"), "polomka" : "Shassi" }
```

И, в заключение, полёт:

```
> db.polet.insert({id_samolet: ObjectId("5eff4b7ad82b8666975964d3"), id_reys: ObjectId("5eff58a5d82b8666975964d8"), "data_vyleta" : "2020-03-14", "vremya_vyleta" : "10:42:57", "data_prileta" : "2020-03-14", "vremya_prileta" : "12:06:17", "kolichestvo_prodannych_biletov" : 167, "kolichestvo_soverchennych_reysov" : 104})
WriteResult({ "nInserted" : 1 })
> db.polet.insert({id_samolet: ObjectId("5eff4f65d82b8666975964d4"), id_reys: ObjectId("5eff58bdd82b8666975964d9"), "data_vyleta" : "2020-03-15", "vremya_vyleta" : "15:56:41", "data_prileta" : "2020-03-15", "vremya_prileta" : "16:56:13", "kolichestvo_prodannych_biletov" : 137, "kolichestvo_soverchennych_reysov" : 18})
WriteResult({ "nInserted" : 1 })
> db.polet.insert({id_samolet: ObjectId("5eff4f6bd82b8666975964d5"), id_reys: ObjectId("5eff58c4d82b8666975964da"), "data_vyleta" : "2020-03-16", "vremya_vyleta" : "23:18:18", "data_prileta" : "2020-03-17", "vremya_prileta" : "00:16:29", "kolichestvo_prodannych_biletov" : 124, "kolichestvo_soverchennych_reysov" : 19})
WriteResult({ "nInserted" : 1 })
> db.polet.insert({id_samolet: ObjectId("5eff4f70d82b8666975964d6"), id_reys: ObjectId("5eff58c9d82b8666975964db"), "data_vyleta" : "2020-03-21", "vremya_vyleta" : "23:56:37", "data_prileta" : "2020-03-23", "vremya_prileta" : "01:45:27", "kolichestvo_prodannych_biletov" : 144, "kolichestvo_soverchennych_reysov" : 27})
WriteResult({ "nInserted" : 1 })
> db.polet.insert({id_samolet: ObjectId("5eff4f70d82b8666975964d7"), id_reys: ObjectId("5eff58cfd82b8666975964dc"), "data_vyleta" : "2020-03-19", "vremya_vyleta" : "19:00:01", "data_prileta" : "2020-03-19", "vremya_prileta" : "19:13:38", "kolichestvo_prodannych_biletov" : 132, "kolichestvo_soverchennych_reysov" : 98})
WriteResult({ "nInserted" : 1 })
> db.polet.find()
{ "_id" : ObjectId("5eff7c997d954ddb96164bad"), "id_samolet" : ObjectId("5eff4b7ad82b8666975964d3"), "id_reys" : ObjectId("5eff58a5d82b8666975964d8"), "data_vyleta" : "2020-03-14", "vremya_vyleta" : "10:42:57", "data_prileta" : "2020-03-14", "vremya_prileta" : "12:06:17", "kolichestvo_prodannych_biletov" : 167, "kolichestvo_soverchennych_reysov" : 104 }
{ "_id" : ObjectId("5eff7cb07d954ddb96164bae"), "id_samolet" : ObjectId("5eff4f65d82b8666975964d4"), "id_reys" : ObjectId("5eff58bdd82b8666975964d9"), "data_vyleta" : "2020-03-15", "vremya_vyleta" : "15:56:41", "data_prileta" : "2020-03-15", "vremya_prileta" : "16:56:13", "kolichestvo_prodannych_biletov" : 137, "kolichestvo_soverchennych_reysov" : 18 }
{ "_id" : ObjectId("5eff7cbc7d954ddb96164baf"), "id_samolet" : ObjectId("5eff4f6bd82b8666975964d5"), "id_reys" : ObjectId("5eff58c4d82b8666975964da"), "data_vyleta" : "2020-03-16", "vremya_vyleta" : "23:18:18", "data_prileta" : "2020-03-17", "vremya_prileta" : "00:16:29", "kolichestvo_prodannych_biletov" : 124, "kolichestvo_soverchennych_reysov" : 19 }
{ "_id" : ObjectId("5eff7cc77d954ddb96164bb0"), "id_samolet" : ObjectId("5eff4f70d82b8666975964d6"), "id_reys" : ObjectId("5eff58c9d82b8666975964db"), "data_vyleta" : "2020-03-21", "vremya_vyleta" : "23:56:37", "data_prileta" : "2020-03-23", "vremya_prileta" : "01:45:27", "kolichestvo_prodannych_biletov" : 144, "kolichestvo_soverchennych_reysov" : 27 }
{ "_id" : ObjectId("5eff7cd27d954ddb96164bb1"), "id_samolet" : ObjectId("5eff4f70d82b8666975964d7"), "id_reys" : ObjectId("5eff58cfd82b8666975964dc"), "data_vyleta" : "2020-03-19", "vremya_vyleta" : "19:00:01", "data_prileta" : "2020-03-19", "vremya_prileta" : "19:13:38", "kolichestvo_prodannych_biletov" : 132, "kolichestvo_soverchennych_reysov" : 98 }
```

Теперь приступим к запросам, будем получать результаты из нашей модели!

1. Найдём только те полёты, которые случились 14 марта 2020:

db.polet.find({data_vyleta: '2020-03-14'}, {_id: 0});

```
> db.polet.find({data_vyleta: '2020-03-14'}, {_id: 0});
{ "id_samolet" : ObjectId("5eff4b7ad82b8666975964d3"), "id_reys" : ObjectId("5eff58a5d82b8666975964d8"), "data_vyleta" : "2020-03-14", "vremya_vyleta" : "10:42:57", "data_prileta" : "2020-03-14", "vremya_prileta" : "12:06:17", "kolichestvo_prodannych_biletov" : 167, "kolichestvo_soverchennych_reysov" : 104 }
```

2. Найдём только те id (бывшие “позывные”) самолёта, которые принадлежат модели «Боинг»:

db.samolet.find({tip: "Boeing"}, {_id: 1});

```
> db.samolet.find({tip: "Boeing"}, {_id: 1});
{ "_id" : ObjectId("5eff4b7ad82b8666975964d3") }
{ "_id" : ObjectId("5eff4f6bd82b8666975964d5") }
{ "_id" : ObjectId("5eff4f74d82b8666975964d7") }
```

3. Посчитаем, сколько самолётов в ремонте с причиной поломки «шасси»:

db.remont.count({polomka : 'Shassi'})

```
> db.remont.count({polomka : 'Shassi'})
1
```

4. Показать количество проданных билетов на рейс, который летал более 20 раз с сортировкой:

db.polet.find({kolichestvo_soverchennych_reysov: {$gt : 20}}, {kolichestvo_prodannych_biletov: 1}).sort({kolichestvo_prodannych_biletov: 1})

```
> db.polet.find({kolichestvo_soverchennych_reysov: {$gt : 20}}, {kolichestvo_prodannych_biletov: 1}).sort({kolichestvo_p
rodannych_biletov: 1})
{ "_id" : ObjectId("5eff7cd27d954ddb96164bb1"), "kolichestvo_prodannych_biletov" : 132 }
{ "_id" : ObjectId("5eff7cc77d954ddb96164bb0"), "kolichestvo_prodannych_biletov" : 144 }
{ "_id" : ObjectId("5eff7c997d954ddb96164bad"), "kolichestvo_prodannych_biletov" : 167 }
```

5. Показать только те позывные и код их полёта, у которых количество проданных билетов на последний рейс превышает 120 и общее количество рейсов больше 50:

db.polet.find({kolichestvo_soverchennych_reysov: {$gt : 50}, kolichestvo_prodannych_biletov: {$gt : 120}}, {_id: 1, id_samolet: 1});

```
> db.polet.find({kolichestvo_soverchennych_reysov: {$gt : 50}, kolichestvo_prodannych_biletov:  {$gt : 120}}, {_id: 1, id
_samolet: 1});
{ "_id" : ObjectId("5eff7c997d954ddb96164bad"), "id_samolet" : ObjectId("5eff4b7ad82b8666975964d3") }
{ "_id" : ObjectId("5eff7cd27d954ddb96164bb1"), "id_samolet" : ObjectId("5eff4f70d82b8666975964d7") }
```

6. Показать количество самолётов определённого типа модели (в нашем случае «боинга»)

db.samolet.find({tip: "Boeing"}).count()

```
> db.samolet.find({tip: "Boeing"}).count()
3
```

7. Найдём длину в символах у выбранных типов модели:

db.samolet.aggregate([{ $project: { "tip": 1, "length": { $strLenCP: "$tip" } } }])

```
> db.samolet.aggregate(
... [
... {
... $project: {
... "tip": 1,
... "length": { $strLenCP: "$tip" }
... }
... }
... ]
... )
{ "_id" : ObjectId("5eff4b7ad82b8666975964d3"), "tip" : "Boeing", "length" : 6 }
{ "_id" : ObjectId("5eff4f65d82b8666975964d4"), "tip" : "Airbus", "length" : 6 }
{ "_id" : ObjectId("5eff4f6bd82b8666975964d5"), "tip" : "Boeing", "length" : 6 }
{ "_id" : ObjectId("5eff4f70d82b8666975964d6"), "tip" : "Airbus", "length" : 6 }
{ "_id" : ObjectId("5eff4f74d82b8666975964d7"), "tip" : "Boeing", "length" : 6 }
>
```

8. Агрегирование результатов с помощью MapReduce:

> function map(){ emit(this.tip, 1); }

> function reduce(key, values) { var sum = 0; for(var i in values) { sum += values[i]; } return sum; }

> db.samolet.mapReduce(map, reduce,{out:"tip"})

```
> function map(){ emit(this.tip, 1); }
> function reduce(key, values) { var sum = 0; for(var i in values) { sum += values[i]; } return sum; }
> db.samolet.mapReduce(map, reduce,{out:"tip"})
{
        "result" : "tip",
        "timeMillis" : 65,
        "counts" : {
                "input" : 5,
                "emit" : 5,
                "reduce" : 2,
                "output" : 2
        },
        "ok" : 1
}
```

Выводы:

В ходе выполнения данной лабораторной работы мы познакомились с Mongodb. Была уменьшена первоначальная база данных, были выполнены запросы к новой модели на основе старых.