Universidade do Minho
Escola de Engenharia

Mestrado em Bioinformática
2023/2024

BASES DE DADOS NOSQL

Trabalho realizado por:

Armindo Machado- PG52170

Duarte Velho- PG53481

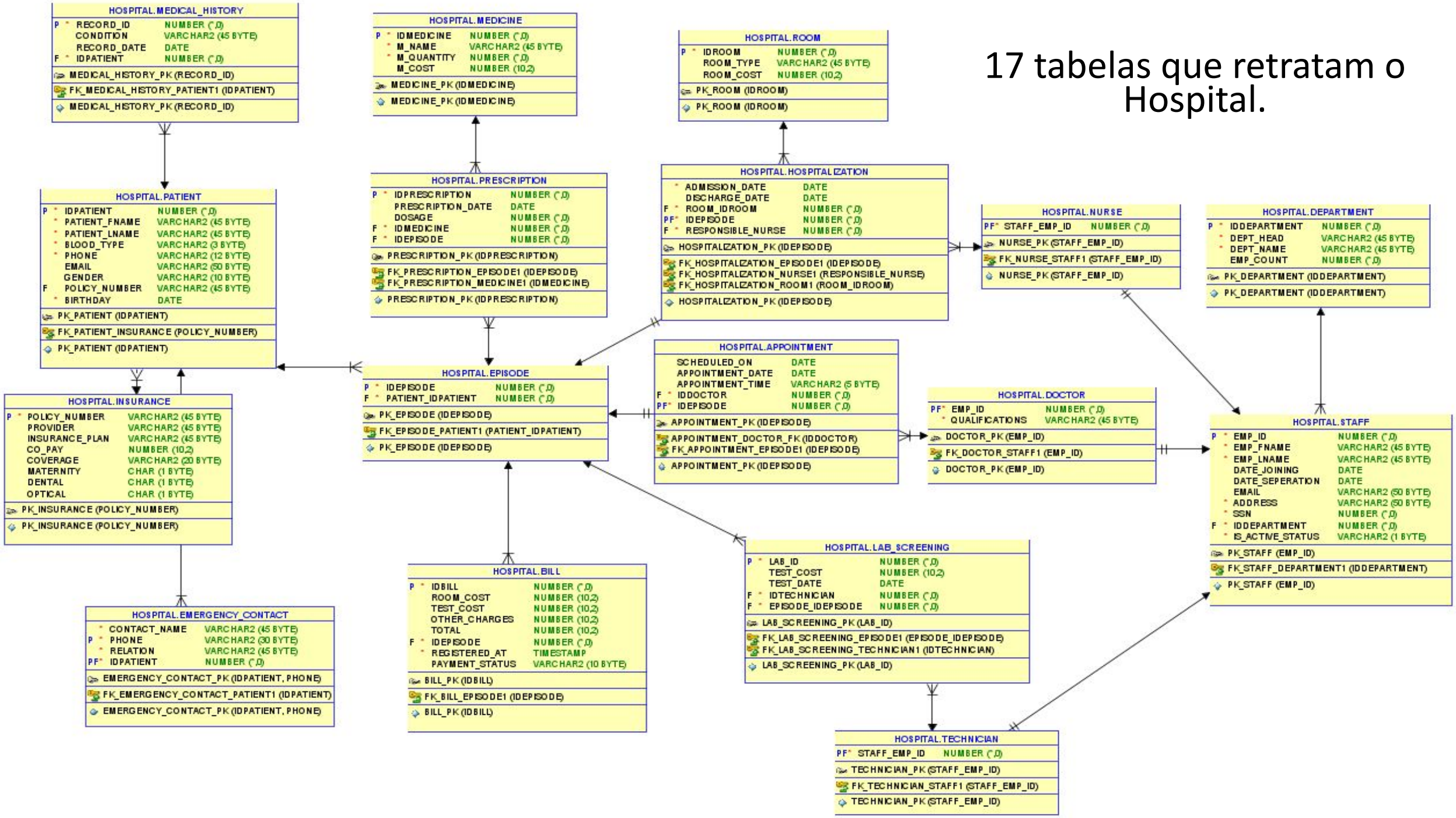
Mariana Oliveira- PG52648

Ricardo Oliveira- PG53501

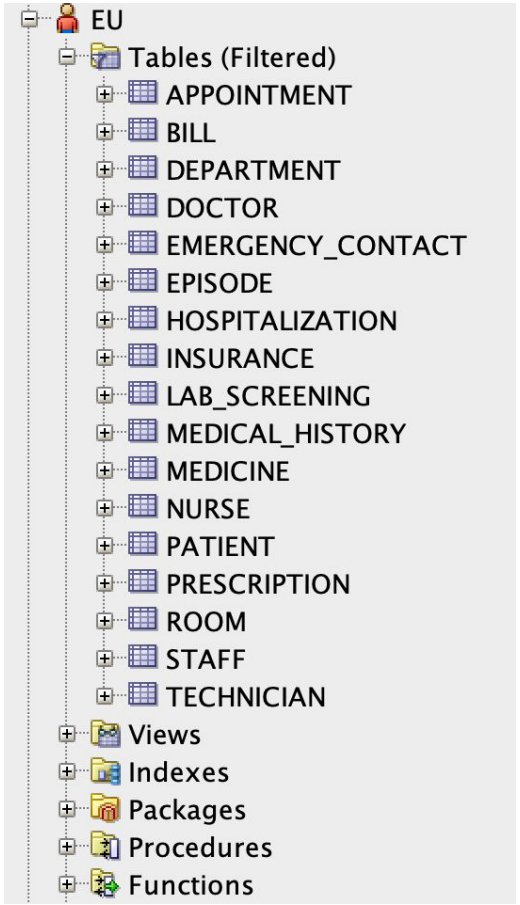
Rodrigo Esperança- PG50923

Objetivo principal:

Explorar diferentes paradigmas de base de dados, tendo como referências dois tipos de modelo, um modelo relacional e dois modelo não relacional.


HOSPITAL.MEDICAL_HISTORY
P * RECORD_ID NUMBER (*,0)
CONDITION VARCHAR2 (45 BYTE)
RECORD_DATE DATE
F * IDPATIENT NUMBER (*,0)
MEDICAL_HISTORY_PK (RECORD_ID)
FK_MEDICAL_HISTORY_PATIENT1 (IDPATIENT)
MEDICAL_HISTORY_PK (RECORD_ID)
HOSPITAL.MEDICINE
P * IDMEDICINE NUMBER (*,0)
* M_NAME VARCHAR2 (45 BYTE)
* M_QUANTITY NUMBER (*,0)
M_COST NUMBER (10,2)
MEDICINE_PK (IDMEDICINE)
MEDICINE_PK (IDMEDICINE)
HOSPITAL.ROOM
P * IDROOM NUMBER (*,0)
ROOM_TYPE VARCHAR2 (45 BYTE)
ROOM_COST NUMBER (10,2)
PK_ROOM (IDROOM)
PK_ROOM (IDROOM)
17 tabelas que retratam o Hospital.
HOSPITAL.PATIENT
P * IDPATIENT NUMBER (*,0)
* PATIENT_FNAME VARCHAR2 (45 BYTE)
* PATIENT_LNAME VARCHAR2 (45 BYTE)
* BLOOD_TYPE VARCHAR2 (3 BYTE)
* PHONE VARCHAR2 (12 BYTE)
EMAIL VARCHAR2 (50 BYTE)
GENDER VARCHAR2 (10 BYTE)
F POLICY_NUMBER VARCHAR2 (45 BYTE)
* BIRTHDAY DATE
PK_PATIENT (IDPATIENT)
FK_PATIENT_INSURANCE (POLICY_NUMBER)
PK_PATIENT (IDPATIENT)
HOSPITAL.PRESCRIPTION
P * IDPRESCRIPTION NUMBER (*,0)
PRESCRIPTION_DATE DATE
DOSAGE NUMBER (*,0)
F * IDMEDICINE NUMBER (*,0)
F * IDEPISODE NUMBER (*,0)
PRESCRIPTION_PK (IDPRESCRIPTION)
FK_PRESCRIPTION_EPISODE1 (IDEPISODE)
FK_PRESCRIPTION_MEDICINE1 (IDMEDICINE)
PRESCRIPTION_PK (IDPRESCRIPTION)
HOSPITAL.HOSPITALIZATION
* ADMISSION_DATE DATE
DISCHARGE_DATE DATE
F * ROOM_IDROOM NUMBER (*,0)
PF* IDEPISODE NUMBER (*,0)
F * RESPONSIBLE_NURSE NUMBER (*,0)
HOSPITALIZATION_PK (IDEPISODE)
FK_HOSPITALIZATION_EPISODE1 (IDEPISODE)
FK_HOSPITALIZATION_NURSE1 (RESPONSIBLE_NURSE)
FK_HOSPITALIZATION_ROOM1 (ROOM_IDROOM)
HOSPITALIZATION_PK (IDEPISODE)
HOSPITAL.NURSE
PF* STAFF_EMP_ID NUMBER (*,0)
NURSE_PK (STAFF_EMP_ID)
FK_NURSE_STAFF1 (STAFF_EMP_ID)
NURSE_PK (STAFF_EMP_ID)
HOSPITAL.DEPARTMENT
P * IDDEPARTMENT NUMBER (*,0)
* DEPT_HEAD VARCHAR2 (45 BYTE)
* DEPT_NAME VARCHAR2 (45 BYTE)
EMP_COUNT NUMBER (*,0)
PK_DEPARTMENT (IDDEPARTMENT)
PK_DEPARTMENT (IDDEPARTMENT)
HOSPITAL.INSURANCE
P * POLICY_NUMBER VARCHAR2 (45 BYTE)
PROVIDER VARCHAR2 (45 BYTE)
INSURANCE_PLAN VARCHAR2 (45 BYTE)
CO_PAY NUMBER (10,2)
COVERAGE VARCHAR2 (20 BYTE)
MATERNITY CHAR (1 BYTE)
DENTAL CHAR (1 BYTE)
OPTICAL CHAR (1 BYTE)
PK_INSURANCE (POLICY_NUMBER)
PK_INSURANCE (POLICY_NUMBER)
HOSPITAL.EPISODE
P * IDEPISODE NUMBER (*,0)
F * PATIENT_IDPATIENT NUMBER (*,0)
PK_EPISODE (IDEPISODE)
FK_EPISODE_PATIENT1 (PATIENT_IDPATIENT)
PK_EPISODE (IDEPISODE)
HOSPITAL.APPOINTMENT
SCHEDULED_ON DATE
APPOINTMENT_DATE DATE
APPOINTMENT_TIME VARCHAR2 (5 BYTE)
F * IDDOCTOR NUMBER (*,0)
PF* IDEPISODE NUMBER (*,0)
APPOINTMENT_PK (IDEPISODE)
APPOINTMENT_DOCTOR_FK (IDDOCTOR)
FK_APPOINTMENT_EPISODE1 (IDEPISODE)
APPOINTMENT_PK (IDEPISODE)
HOSPITAL.DOCTOR
PF* EMP_ID NUMBER (*,0)
* QUALIFICATIONS VARCHAR2 (45 BYTE)
DOCTOR_PK (EMP_ID)
FK_DOCTOR_STAFF1 (EMP_ID)
DOCTOR_PK (EMP_ID)
HOSPITAL.STAFF
P * EMP_ID NUMBER (*,0)
* EMP_FNAME VARCHAR2 (45 BYTE)
* EMP_LNAME VARCHAR2 (45 BYTE)
DATE_JOINING DATE
DATE_SEPERATION DATE
EMAIL VARCHAR2 (50 BYTE)
* ADDRESS VARCHAR2 (50 BYTE)
* SSN NUMBER (*,0)
F * IDDEPARTMENT NUMBER (*,0)
* IS_ACTIVE_STATUS VARCHAR2 (1 BYTE)
PK_STAFF (EMP_ID)
FK_STAFF_DEPARTMENT1 (IDDEPARTMENT)
PK_STAFF (EMP_ID)
HOSPITAL.EMERGENCY_CONTACT
* CONTACT_NAME VARCHAR2 (45 BYTE)
P * PHONE VARCHAR2 (30 BYTE)
* RELATION VARCHAR2 (45 BYTE)
PF* IDPATIENT NUMBER (*,0)
EMERGENCY_CONTACT_PK (IDPATIENT, PHONE)
FK_EMERGENCY_CONTACT_PATIENT1 (IDPATIENT)
EMERGENCY_CONTACT_PK (IDPATIENT, PHONE)
HOSPITAL.BILL
P * IDBILL NUMBER (*,0)
ROOM_COST NUMBER (10,2)
TEST_COST NUMBER (10,2)
OTHER_CHARGES NUMBER (10,2)
TOTAL NUMBER (10,2)
F * IDEPISODE NUMBER (*,0)
* REGISTERED_AT TIMESTAMP
PAYMENT_STATUS VARCHAR2 (10 BYTE)
BILL_PK (IDBILL)
FK_BILL_EPISODE1 (IDEPISODE)
BILL_PK (IDBILL)
HOSPITAL.LAB_SCREENING
P * LAB_ID NUMBER (*,0)
TEST_COST NUMBER (10,2)
TEST_DATE DATE
F * IDTECHNICIAN NUMBER (*,0)
F * EPISODE_IDEPISODE NUMBER (*,0)
LAB_SCREENING_PK (LAB_ID)
FK_LAB_SCREENING_EPISODE1 (EPISODE_IDEPISODE)
FK_LAB_SCREENING_TECHNICIAN1 (IDTECHNICIAN)
LAB_SCREENING_PK (LAB_ID)
HOSPITAL.TECHNICIAN
PF* STAFF_EMP_ID NUMBER (*,0)
TECHNICIAN_PK (STAFF_EMP_ID)
FK_TECHNICIAN_STAFF1 (STAFF_EMP_ID)
TECHNICIAN_PK (STAFF_EMP_ID)

1º Abordagem- Oracle




```
SELECT * FROM PATIENT;
```

IDPATIENT	PATIENT_FNAME	PATIENT_LNAME	BLO	PHONE	EMAIL
1	John	Doe	A+	123-456-7890	john.doe@example.com
2	Jane	Smith	O-	987-654-3210	jane.smith@example.com
3	Michael	Johnson	B+	567-890-1234	michael.johnson@example.com
4	Emily	Brown	AB-	789-012-3456	emily.brown@example.com
5	William	Martinez	A-	234-567-8901	william.martinez@example.com
6	Sophia	Garcia	O+	890-123-4567	sophia.garcia@example.com
7	James	Lopez	B-	456-789-0123	james.lopez@example.com
8	Olivia	Lee	AB+	901-234-5678	olivia.lee@example.com
9	Benjamin	Gonzalez	O-	678-901-2345	benjamin.gonzalez@example.com
10	Emma	Perez	A+	345-678-9012	emma.perez@example.com
11	Jacob	Rodriguez	B+	123-123-1234	jacob.rodriguez@example.com

2º Abordagem - MongoDB

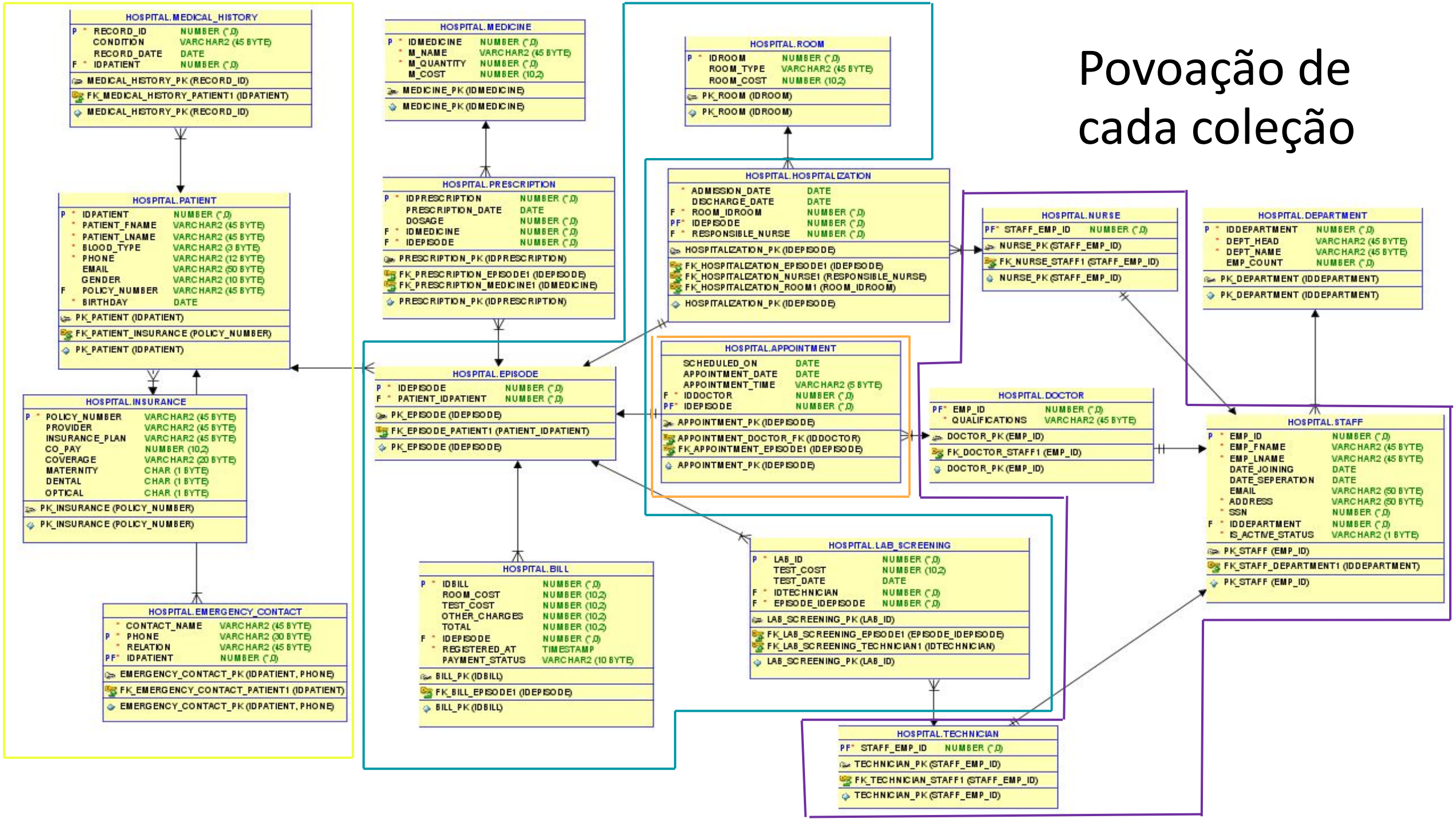
Patients

Appointments

Services

Staff

Povoação de cada coleção



QUERY'S de consulta

Número de pacientes

```
def count_patients():
    client = MongoClient('mongodb://localhost:27017/')
    db = client['Hospital']
    count = db['Patients'].count_documents({})
    print(f"Número total de pacientes: {count}")
    client.close()

count_patients()
```

```
Número total de pacientes: 378
```

Lista dos pacientes por tipo sanguíneo

```
Quantidade de pacientes com tipo sanguíneo O+: 36
Nome: Amelia Tran, Tipo Sanguíneo: O+
Nome: Amelia Tran, Tipo Sanguíneo: O+
Nome: Sophia Garcia, Tipo Sanguíneo: O+
Nome: Mia Gomez, Tipo Sanguíneo: O+
Nome: Emma Flores, Tipo Sanguíneo: O+
Nome: Charlotte Ngo, Tipo Sanguíneo: O+
Nome: Sophia Nguyen, Tipo Sanguíneo: O+
```

Contar o número de consultas

```
Total number of appointments: 99
```

Pacientes por género

```
Quantidade de pacientes do sexo masculino: 138
Quantidade de pacientes do sexo feminino: 132
Nome: Amelia Tran, Sexo: Female
Nome: Amelia Tran, Sexo: Female
Nome: Jane Smith, Sexo: Female
Nome: Emily Brown, Sexo: Female
Nome: Sophia Garcia, Sexo: Female
Nome: Olivia Lee, Sexo: Female
```

Index:

Na coleção 'Patients', os índices são criados para facilitar buscas por data de nascimento e gênero e para o número de seguro

Na coleção 'Appointments', os índices são estabelecidos para a data agendada e uma combinação do médico e da data da consulta.

Na coleção 'Services' utiliza índices para episódios de tratamento e identificação de faturas

Na coleção 'Staff', são criados índices para o departamento ao qual pertencem e o tipo de função que exercem

```
# Conectar ao MongoDB
client = MongoClient('mongodb://localhost:27017/')
db = client['Hospital']

# Selecionar coleções
collection_p = db['Patients']
collection_a = db['Appointments']
collection_s = db['Services']
collection_sa = db['Staff']

# Criar índices nas coleções
# Índices para a coleção Patients
collection_p.create_index([("idpatient", 1)], unique=True)  # Índice único para identificação do paciente
collection_p.create_index([("BIRTHDAY", 1), ("gender", 1)])  # Índice composto para buscas por data de nascimento e gênero
collection_p.create_index([("policy_number", 1)])  # Índice para número de seguro

# Índices para a coleção Appointments
collection_a.create_index([("scheduled_on", 1)])  # Índice para data agendada
collection_a.create_index([("iddoctor", 1), ("appointment_date", 1)])  # Índice composto para consultas por médico e data

# Índices para a coleção Services
collection_s.create_index([("idepisode", 1)])  # Índice para identificação do episódio
collection_s.create_index([("idbill", 1)])  # Índice para identificação da fatura

# Índices para a coleção Staff
collection_sa.create_index([("_id", 1)], unique=True)  # Índice único para identificação do funcionário
collection_sa.create_index([("departmentId", 1)])  # Índice para identificação do departamento
collection_sa.create_index([("type", 1)])  # Índice para tipo de funcionário (nurse, doctor, technician)

# Fechar conexão
client.close()
```

Views/Agregação:

Demonstrar se existem médicos que também são pacientes

```
aggregate_doctor_patients()

[47] ✓ 0.0s

... Não há nenhum médico que seja paciente
```

Consultar as receitas totais do hospital


```
{'idepisode': 2, 'patient_idpatient': 2, 'bill': {'idbill': 48, 'room_cost': 100.0, 'test_cost': 198.34, 'other_charges': 9905.0, 'total': 10203.34, 'registered_at': datetime.datetime(
{'idepisode': 3, 'patient_idpatient': 3, 'bill': {'idbill': 49, 'room_cost': 150.0, 'test_cost': 0.0, 'other_charges': 3505.0, 'total': 3655.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 3, 'patient_idpatient': 3, 'bill': {'idbill': 1, 'room_cost': 150.0, 'test_cost': 0.0, 'other_charges': 3505.0, 'total': 3655.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 5, 'patient_idpatient': 5, 'bill': {'idbill': 50, 'room_cost': 250.0, 'test_cost': 0.0, 'other_charges': 7100.0, 'total': 7350.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 5, 'patient_idpatient': 5, 'bill': {'idbill': 2, 'room_cost': 250.0, 'test_cost': 0.0, 'other_charges': 7100.0, 'total': 7350.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 6, 'patient_idpatient': 6, 'bill': {'idbill': 51, 'room_cost': 80.0, 'test_cost': 0.0, 'other_charges': 4490.0, 'total': 4570.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 6, 'patient_idpatient': 6, 'bill': {'idbill': 3, 'room_cost': 80.0, 'test_cost': 0.0, 'other_charges': 4490.0, 'total': 4570.0, 'registered_at': datetime.datetime(2024, 4
{'idepisode': 9, 'patient_idpatient': 9, 'bill': {'idbill': 52, 'room_cost': 200.0, 'test_cost': 0.0, 'other_charges': 7645.0, 'total': 7845.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 9, 'patient_idpatient': 9, 'bill': {'idbill': 4, 'room_cost': 200.0, 'test_cost': 0.0, 'other_charges': 7645.0, 'total': 7845.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 11, 'patient_idpatient': 11, 'bill': {'idbill': 53, 'room_cost': 300.0, 'test_cost': 0.0, 'other_charges': 6030.0, 'total': 6330.0, 'registered_at': datetime.datetime(202
{'idepisode': 11, 'patient_idpatient': 11, 'bill': {'idbill': 5, 'room_cost': 300.0, 'test_cost': 0.0, 'other_charges': 6030.0, 'total': 6330.0, 'registered_at': datetime.datetime(2024
{'idepisode': 14, 'patient_idpatient': 14, 'bill': {'idbill': 54, 'room_cost': 70.0, 'test_cost': 0.0, 'other_charges': 1910.0, 'total': 1980.0, 'registered_at': datetime.datetime(2024
{'idepisode': 14, 'patient_idpatient': 14, 'bill': {'idbill': 6, 'room_cost': 70.0, 'test_cost': 0.0, 'other_charges': 1910.0, 'total': 1980.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 25, 'patient_idpatient': 25, 'bill': {'idbill': 55, 'room_cost': 180.0, 'test_cost': 0.0, 'other_charges': 370.0, 'total': 550.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 25, 'patient_idpatient': 25, 'bill': {'idbill': 7, 'room_cost': 180.0, 'test_cost': 0.0, 'other_charges': 370.0, 'total': 550.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 27, 'patient_idpatient': 27, 'bill': {'idbill': 56, 'room_cost': 500.0, 'test_cost': 0.0, 'other_charges': 7870.0, 'total': 8370.0, 'registered_at': datetime.datetime(202
{'idepisode': 27, 'patient_idpatient': 27, 'bill': {'idbill': 8, 'room_cost': 500.0, 'test_cost': 0.0, 'other_charges': 7870.0, 'total': 8370.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 28, 'patient_idpatient': 28, 'bill': {'idbill': 9, 'room_cost': 400.0, 'test_cost': 36.36, 'other_charges': 13060.0, 'total': 13496.36, 'registered_at': datetime.datetime
{'idepisode': 29, 'patient_idpatient': 29, 'bill': {'idbill': 10, 'room_cost': 120.0, 'test_cost': 167.73, 'other_charges': 9805.0, 'total': 10092.73, 'registered_at': datetime.datetim
{'idepisode': 31, 'patient_idpatient': 31, 'bill': {'idbill': 59, 'room_cost': 180.0, 'test_cost': 0.0, 'other_charges': 160.0, 'total': 340.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 31, 'patient_idpatient': 31, 'bill': {'idbill': 11, 'room_cost': 180.0, 'test_cost': 0.0, 'other_charges': 160.0, 'total': 340.0, 'registered_at': datetime.datetime(2024,
{'idepisode': 33, 'patient_idpatient': 33, 'bill': {'idbill': 12, 'room_cost': 280.0, 'test_cost': 268.57, 'other_charges': 10795.0, 'total': 11343.57, 'registered_at': datetime.datetim
{'idepisode': 42, 'patient_idpatient': 42, 'bill': {'idbill': 13, 'room_cost': 90.0, 'test_cost': 64.6, 'other_charges': 7450.0, 'total': 7604.6, 'registered_at': datetime.datetime(202
{'idepisode': 54, 'patient_idpatient': 54, 'bill': {'idbill': 62, 'room_cost': 320.0, 'test_cost': 0.0, 'other_charges': 2880.0, 'total': 3200.0, 'registered_at': datetime.datetime(202
{'idepisode': 54, 'patient_idpatient': 54, 'bill': {'idbill': 14, 'room_cost': 320.0, 'test_cost': 0.0, 'other_charges': 2880.0, 'total': 3200.0, 'registered_at': datetime.datetime(202
...
{'idepisode': 120, 'patient_idpatient': 73, 'bill': {'idbill': 45, 'room_cost': 180.0, 'test_cost': 134.49, 'other_charges': 4210.0, 'total': 4524.49, 'registered_at': datetime.datetim
{'idepisode': 122, 'patient_idpatient': 75, 'bill': {'idbill': 46, 'room_cost': 500.0, 'test_cost': 90.97, 'other_charges': 3530.0, 'total': 4120.97, 'registered_at': datetime.datetime
{'idepisode': 125, 'patient_idpatient': 78, 'bill': {'idbill': 95, 'room_cost': 400.0, 'test_cost': 0.0, 'other_charges': 2730.0, 'total': 3130.0, 'registered_at': datetime.datetime(20
{'idepisode': 125, 'patient_idpatient': 78, 'bill': {'idbill': 47, 'room_cost': 400.0, 'test_cost': 0.0, 'other_charges': 2730.0, 'total': 3130.0, 'registered_at': datetime.datetime(20
```


Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Contar as consultas por médico, incluindo o número de consultas por mês para cada médico


```
{'_id': {'iddoctor': 15, 'month': 5, 'year': 2020}, 'total_appointments': 1, 'doctor_id': 15,
{'_id': {'iddoctor': 83, 'month': 2, 'year': 2023}, 'total_appointments': 1, 'doctor_id': 83,
{'_id': {'iddoctor': 71, 'month': 6, 'year': 2023}, 'total_appointments': 1, 'doctor_id': 71,
{'_id': {'iddoctor': 24, 'month': 2, 'year': 2023}, 'total_appointments': 1, 'doctor_id': 24,
{'_id': {'iddoctor': 66, 'month': 12, 'year': 2015}, 'total_appointments': 1, 'doctor_id': 66,
{'_id': {'iddoctor': 1, 'month': 11, 'year': 2023}, 'total_appointments': 1, 'doctor_id': 1, '
{'_id': {'iddoctor': 14, 'month': 11, 'year': 2023}, 'total_appointments': 1, 'doctor_id': 14,
{'_id': {'iddoctor': 15, 'month': 8, 'year': 2019}, 'total_appointments': 1, 'doctor_id': 15,
```

Views/Agregação:

Demonstrar o número de pacientes por faixa etária

```
            {
                "$bucket": {
                    "groupBy": "$age",
                    "boundaries": [0, 18, 35, 50, 65, 100],
                    "default": "100+",
                    "output": {
                        "count": {"$sum": 1}
                    }
                }
            }
        ]

        result = collection.aggregate(pipeline)

        for age_range in result:
            print(age_range)

        client.close()

    # Chamada da função para fazer a agregação
    aggregate_patients_by_age_range()
```

[53] ✓ 0.0s

```
{'_id': 18, 'count': 75}
{'_id': 35, 'count': 192}
{'_id': 50, 'count': 3}
```

Contar o número de consultas por médico

```
{'_id': 57, 'total_appointments': 2, 'doctor_id': 57, 'doctor_name': 'Ellen', 'doctor_lastname': 'Wright'}
{'_id': 89, 'total_appointments': 6, 'doctor_id': 89, 'doctor_name': 'Rebecca', 'doctor_lastname': 'Reyes'}
{'_id': 56, 'total_appointments': 2, 'doctor_id': 56, 'doctor_name': 'Aaron', 'doctor_lastname': 'Turner'}
{'_id': 83, 'total_appointments': 5, 'doctor_id': 83, 'doctor_name': 'Lee', 'doctor_lastname': 'Collins'}
{'_id': 92, 'total_appointments': 6, 'doctor_id': 92, 'doctor_name': 'Sarah', 'doctor_lastname': 'Boyd'}
{'_id': 85, 'total_appointments': 6, 'doctor_id': 85, 'doctor_name': 'Ashley', 'doctor_lastname': 'Lucas'}
{'_id': 66, 'total_appointments': 5, 'doctor_id': 66, 'doctor_name': 'Christina', 'doctor_lastname': 'Dalton'}
{'_id': 24, 'total_appointments': 4, 'doctor_id': 24, 'doctor_name': 'James', 'doctor_lastname': 'Carpenter'}
{'_id': 14, 'total_appointments': 3, 'doctor_id': 14, 'doctor_name': 'Andrew', 'doctor_lastname': 'Deleon'}
```

Trigger:

MongoDB Atlas

Regista quando um novo documento é inserido

```
exports = function(changeEvent) {
  const fullDocument = changeEvent.fullDocument;
  console.log(`Novo documento inserido com ID: ${fullDocument._id}`);
  // Adicionar qualquer lógica adicional que você precise aqui
};
```

Notificar o paciente sobre os dias que faltam para a consulta

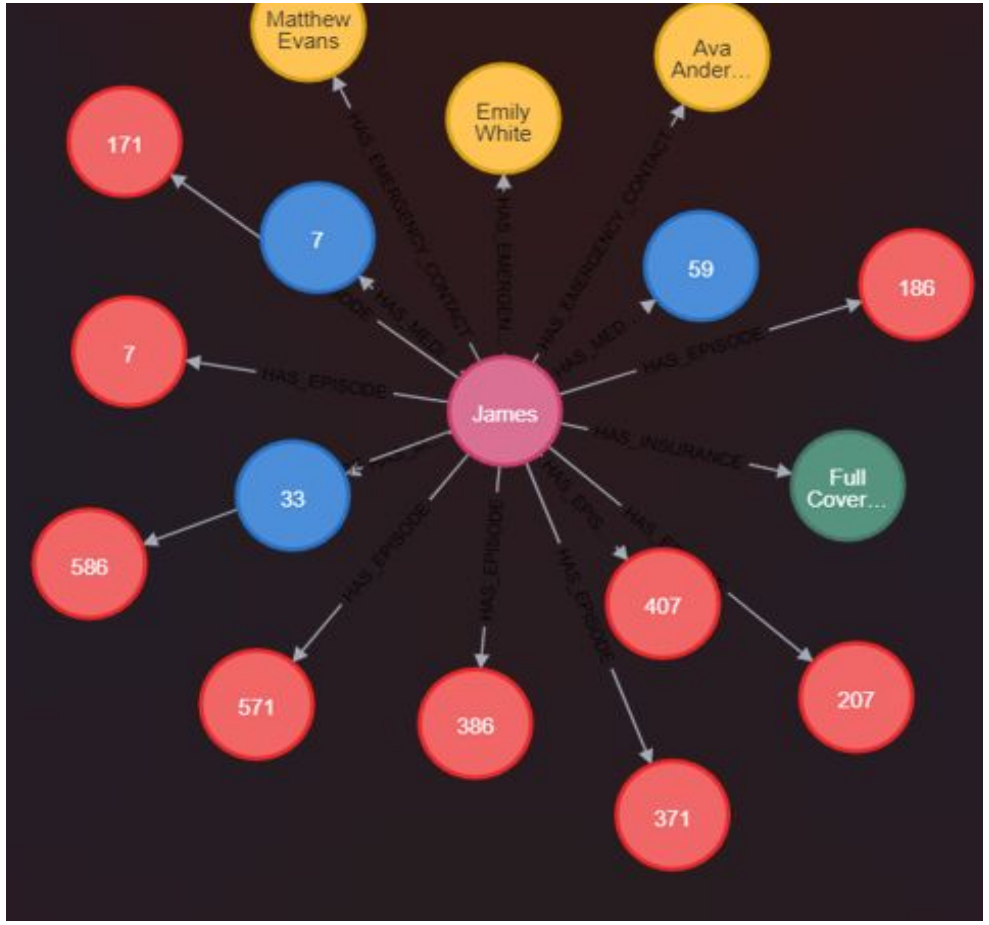
```
exports = async function(changeEvent) {
  const mongodb = context.services.get("mongodb-atlas"); // Verifique o nome do serviço
  const appointmentsCollection = mongodb.db("Hospital").collection("Appointments");
  const patientsCollection = mongodb.db("Hospital").collection("Patients");

  const { fullDocument } = changeEvent;

  if (fullDocument) {
    const appointmentDate = new Date(fullDocument.appointment_date);
    const currentDate = new Date();
    const timeDiff = appointmentDate - currentDate;
    const daysUntilAppointment = Math.ceil(timeDiff / (1000 * 60 * 60 * 24));

    // Encontre o paciente correspondente
    const patient = await patientsCollection.findOne({ idepisode: fullDocument.idepisode });

    if (patient) {
      const sgMail = require('@sendgrid/mail');
      sgMail.setApiKey(context.values.get("SENDGRID_API_KEY"));

      const msg = {
        to: patient.email,
        from: 'seuemail@example.com',
        subject: 'Consulta Marcada',
        text: `Olá ${patient.patient_fname}, sua consulta está marcada para ${fullDocument.appointment_date}. Faltam ${daysUntilAppointment} dias.`,
      };

      await sgMail.send(msg);
    }
  }
};
```

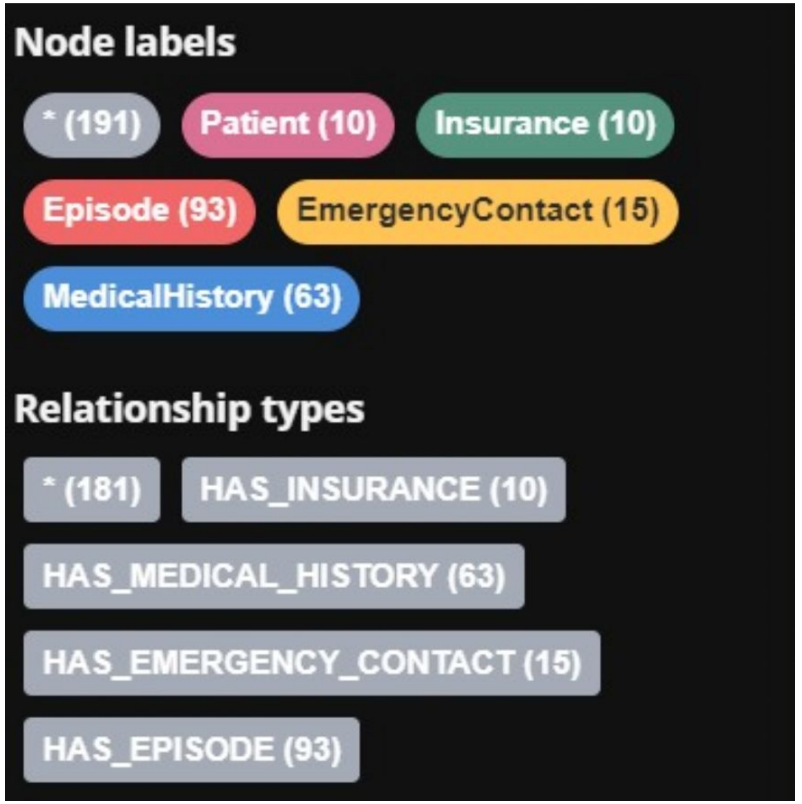
3º Abordagem – Neo4j

Labels:

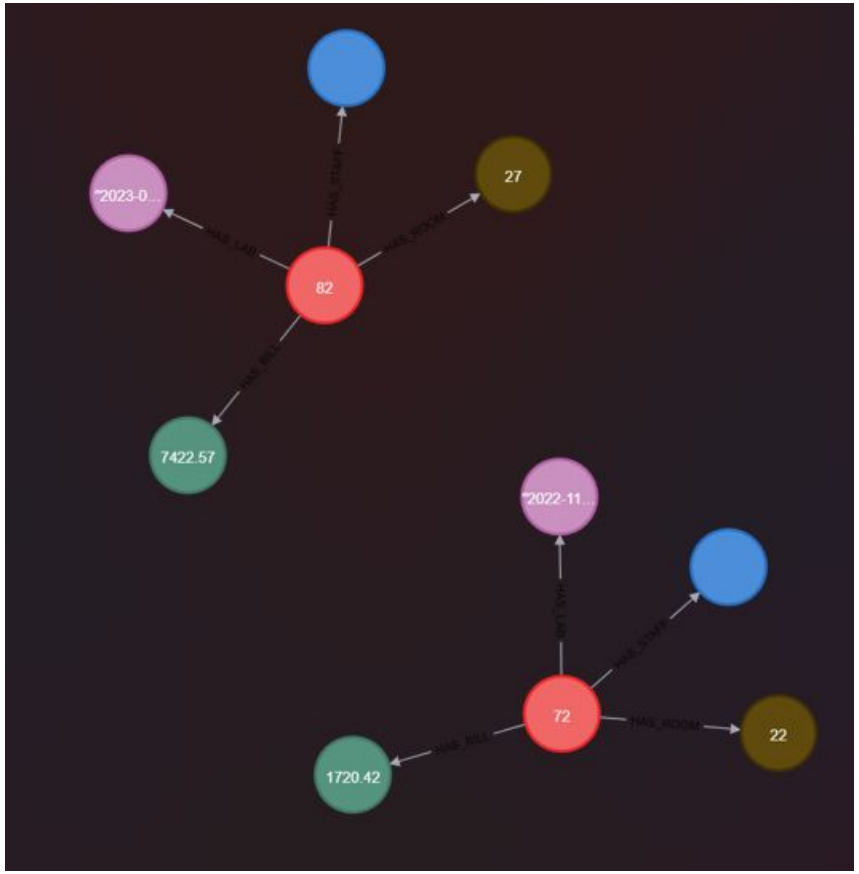
Relacionamentos:

Povoação do Neo4j (patients):





Povoação do Neo4j (episode):



QUERY'S de consulta

Consultar dados de Appointment

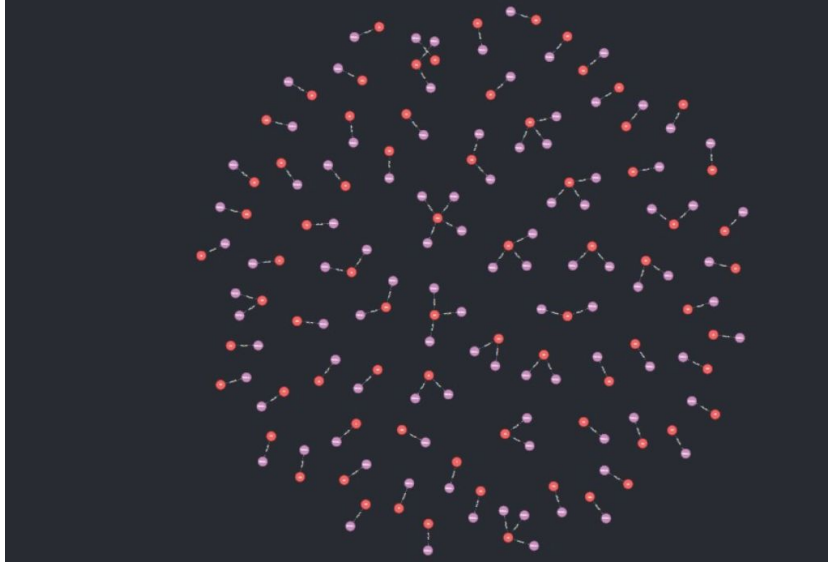
	a.scheduled_on	a.appointment_date	a.appointment_time	a.iddoctor	a.idepisode
1	"2013-11-20T00:00:00"	"2013-12-21T00:00:00"	"13:13"	99	1
2	"2017-10-07T00:00:00"	"2017-11-08T00:00:00"	"16:47"	96	59

Consultar dados de lab_screening

l.lab_id	l.test_cost	l.test_date	l.idtechnician	l.episode_idepisode
1	189.58	"2022-05-24T00:00:00"	43	*null*
2	16.96	"2023-07-27T00:00:00"	46	*null*

Consultar todos os episódios com exames de laboratório realizados

```
MATCH (e:Episode)-[:HAS_LAB]→(l:LabScreening) RETURN e, l;
```

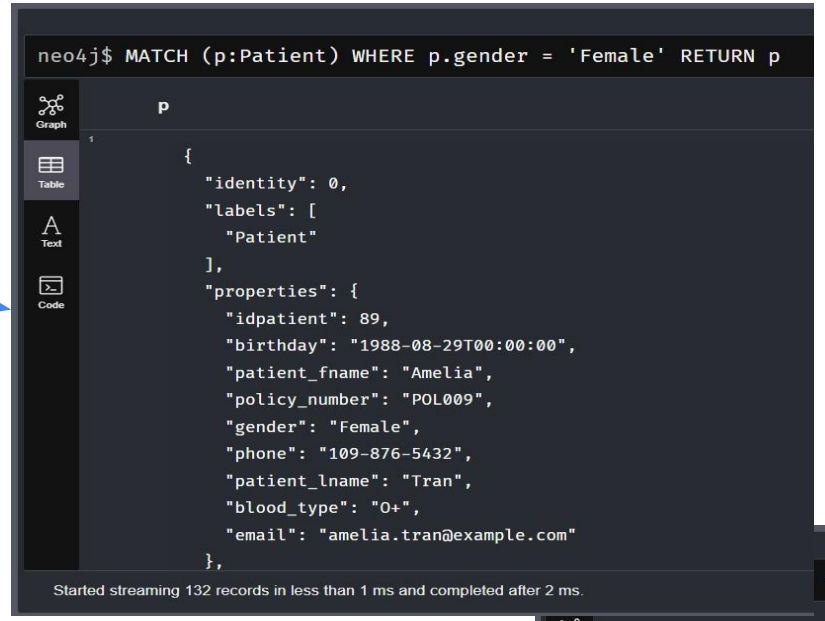


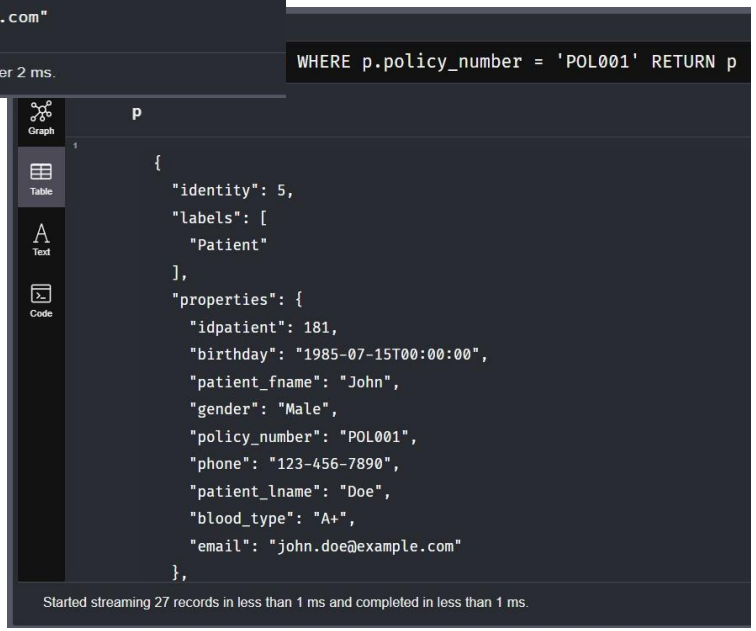
Index

Buscar Pacientes por Data de Nascimento e Gênero

Buscar Paciente por Número de Seguro

Indexação de Contas Hospitalares





Conclusão:

MongoDB

- Esquema muito flexível, mas é complicado fazer uma migração de uma BD relacional para o mongo: obriga a repetir dados (comprometendo a consistência) ou perder dados
- Boa performance triggers, agregações e índices.

Neo4j

- Implica uma preparação muito cuidada do esquema;
- Neo4j não dá erro aquando a inserção de dados repetidos -> atualiza a informação autonomamente;
- O uso de índices não é tão relevante

Universidade do Minho
Escola de Engenharia

Mestrado em Bioinformática
2023/2024

Obrigado pela atenção!