**Centro Universitário UNISATC**

Engenharia de Software 3a fase – Banco de Dados II – Prof.Jorge Luiz da Silva

**TRABALHO FINAL COM BASE EM METODOLOGIAS ATIVAS DE APRENDIZAGEM**

**Projeto de banco de dados para um sistema de Clínica Odontológica**

Ana Carolina Gregório Gonçalves - @anagregorioo

Anna Clara Teixeira de Medeiros - @annaclaratxm

Gabriel Antonin Pascoali - @lugialo

Gabrielle Coelho Silveira - @agabriellecoelho

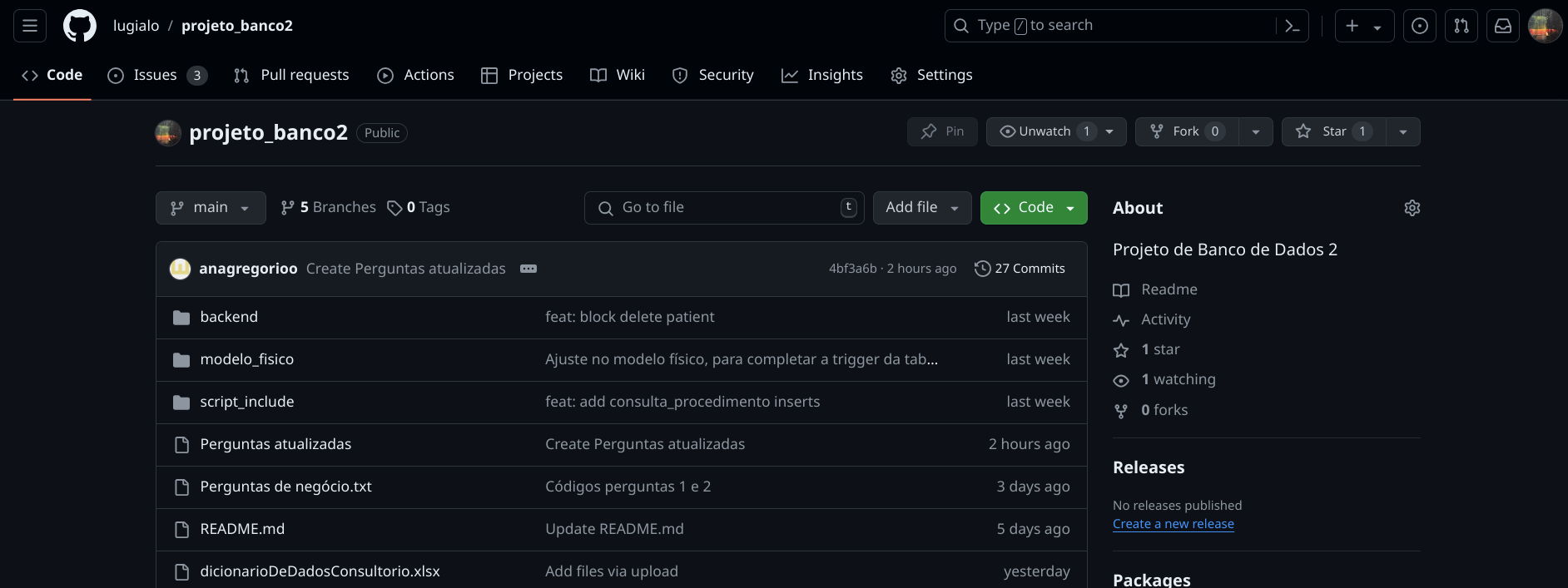
Lourenço Lencina Quevedo - @Lorelq

Vitor Merencio da Luz - @xyzmerencio

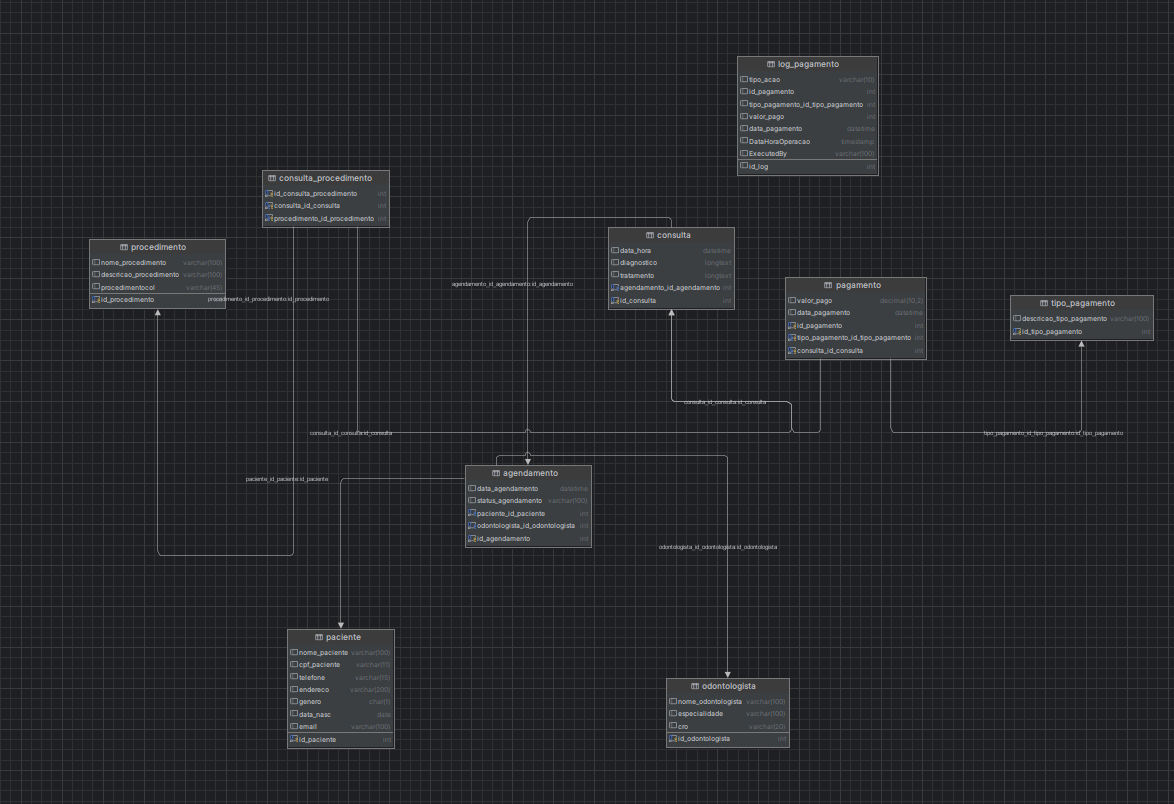
Criciúma, junho de 2024.

**URL do projeto no GitHub**

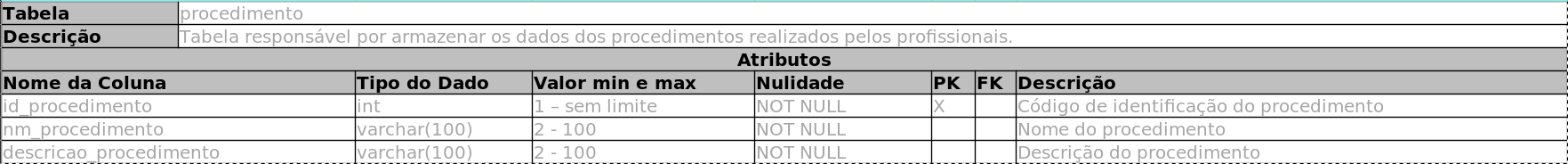
https://github.com/lugialo/projeto\_banco2

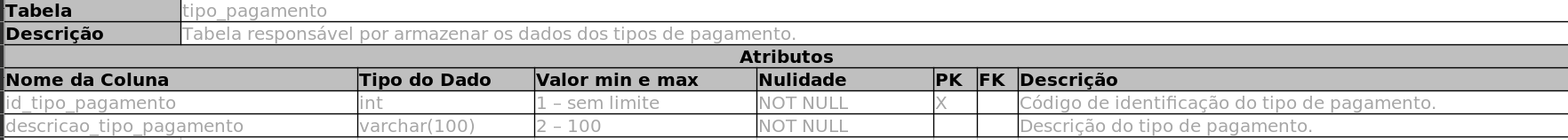
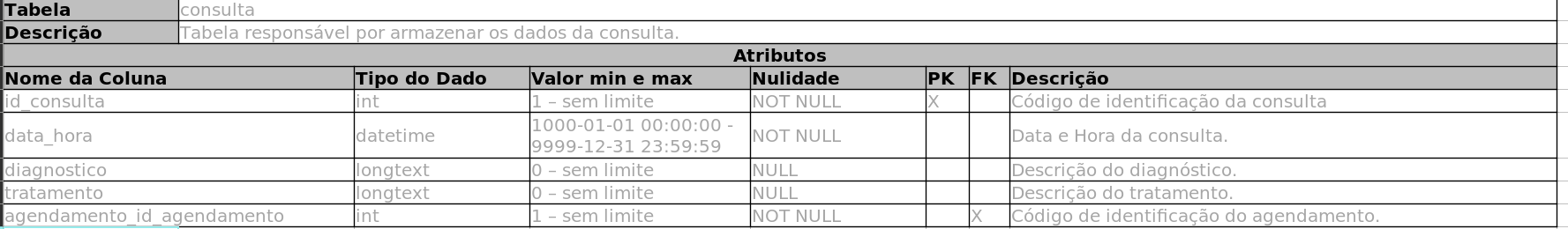
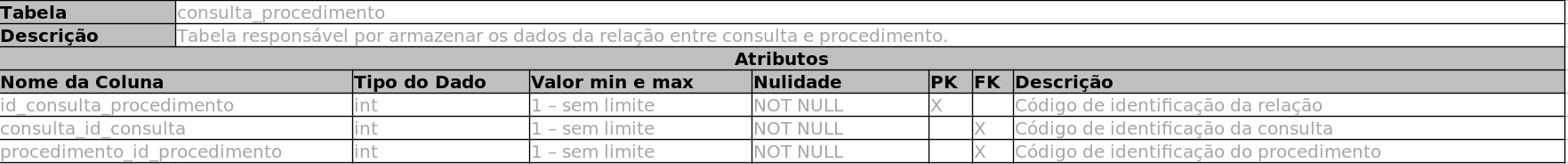
****

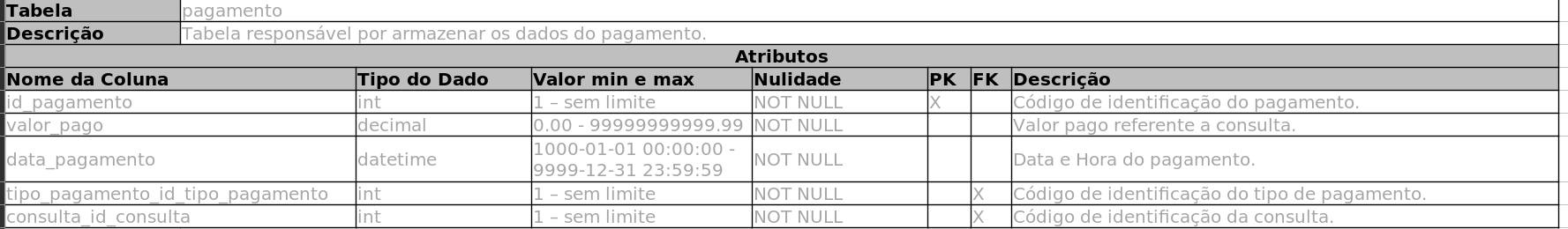
**Modelo ER Físico**

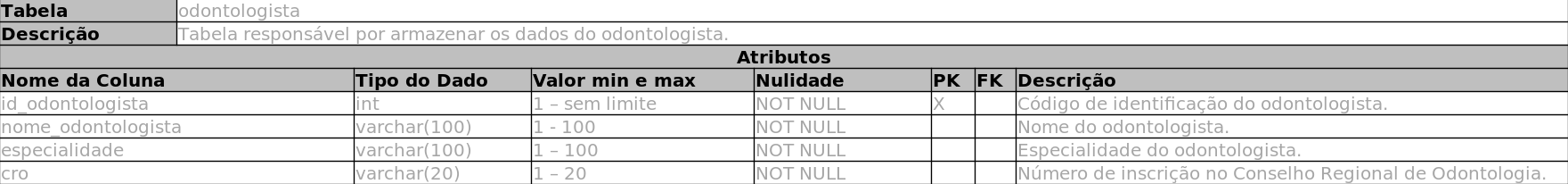
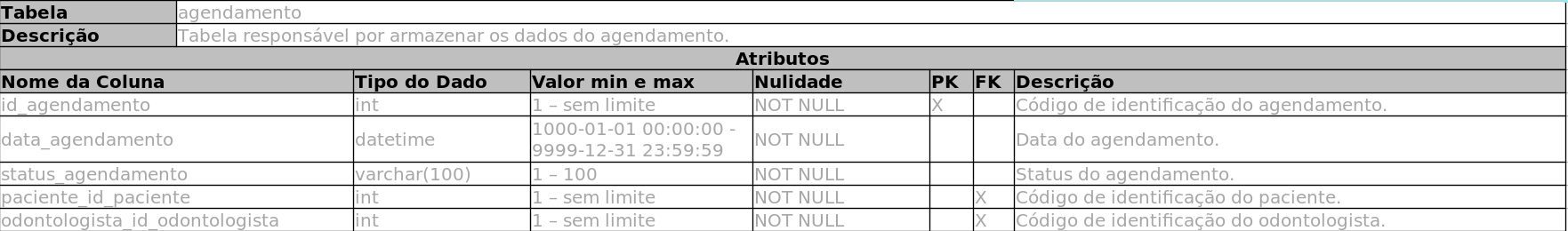
****

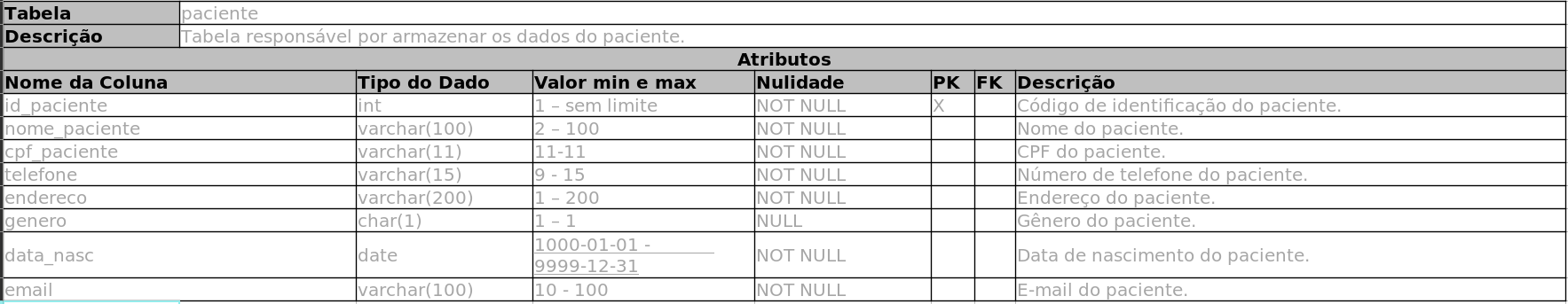
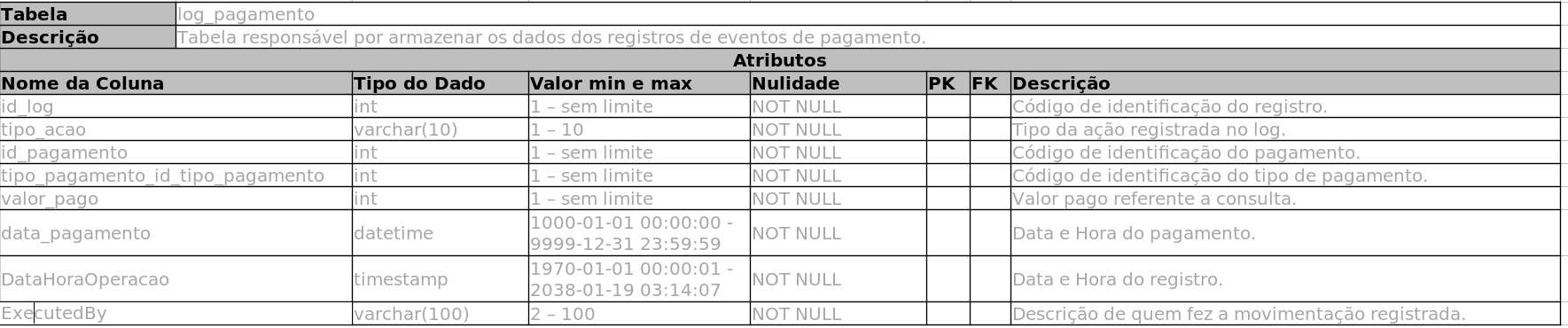
**Dicionário de Dados**











**Script dos comandos DDL para criação do Banco de dados**

create table log\_pagamento

(

id\_log int auto\_increment

primary key,

tipo\_acao varchar(10) not null,

id\_pagamento int not null,

tipo\_pagamento\_id\_tipo\_pagamento int null,

valor\_pago int null,

data\_pagamento datetime null,

DataHoraOperacao timestamp default CURRENT\_TIMESTAMP null,

ExecutedBy varchar(100) null

);

create table odontologista

(

id\_odontologista int auto\_increment

primary key,

nome\_odontologista varchar(100) null,

especialidade varchar(100) null,

cro varchar(20) null

);

create table paciente

(

id\_paciente int auto\_increment

primary key,

nome\_paciente varchar(100) null,

cpf\_paciente varchar(11) null,

telefone varchar(15) null,

endereco varchar(200) null,

genero char null,

data\_nasc date null,

email varchar(100) null

);

create table agendamento

(

id\_agendamento int auto\_increment

primary key,

data\_agendamento datetime null,

status\_agendamento varchar(100) null,

paciente\_id\_paciente int not null,

odontologista\_id\_odontologista int not null,

constraint fk\_agendamento\_odontologista1

foreign key (odontologista\_id\_odontologista) references odontologista (id\_odontologista),

constraint fk\_agendamento\_paciente1

foreign key (paciente\_id\_paciente) references paciente (id\_paciente)

);

create index fk\_agendamento\_odontologista1\_idx

on agendamento (odontologista\_id\_odontologista);

create index fk\_agendamento\_paciente1\_idx

on agendamento (paciente\_id\_paciente);

create table consulta

(

id\_consulta int auto\_increment

primary key,

data\_hora datetime null,

diagnostico longtext null,

tratamento longtext null,

agendamento\_id\_agendamento int not null,

constraint fk\_consulta\_agendamento1

foreign key (agendamento\_id\_agendamento) references agendamento (id\_agendamento)

);

create index fk\_consulta\_agendamento1\_idx

on consulta (agendamento\_id\_agendamento);

create table procedimento

(

id\_procedimento int auto\_increment

primary key,

nome\_procedimento varchar(100) null,

descricao\_procedimento varchar(100) null,

procedimentocol varchar(45) null

);

create table consulta\_procedimento

(

id\_consulta\_procedimento int auto\_increment,

consulta\_id\_consulta int not null,

procedimento\_id\_procedimento int not null,

primary key (id\_consulta\_procedimento, consulta\_id\_consulta, procedimento\_id\_procedimento),

constraint fk\_consulta\_procedimento\_consulta1

foreign key (consulta\_id\_consulta) references consulta (id\_consulta),

constraint fk\_consulta\_procedimento\_procedimento1

foreign key (procedimento\_id\_procedimento) references procedimento (id\_procedimento)

);

create index fk\_consulta\_procedimento\_consulta1\_idx

on consulta\_procedimento (consulta\_id\_consulta);

create index fk\_consulta\_procedimento\_procedimento1\_idx

on consulta\_procedimento (procedimento\_id\_procedimento);

create table tipo\_pagamento

(

id\_tipo\_pagamento int auto\_increment

primary key,

descricao\_tipo\_pagamento varchar(100) null

);

create table pagamento

(

id\_pagamento int auto\_increment,

valor\_pago decimal(10, 2) null,

data\_pagamento datetime null,

tipo\_pagamento\_id\_tipo\_pagamento int not null,

consulta\_id\_consulta int not null,

primary key (id\_pagamento, tipo\_pagamento\_id\_tipo\_pagamento, consulta\_id\_consulta),

constraint fk\_pagamento\_consulta1

foreign key (consulta\_id\_consulta) references consulta (id\_consulta),

constraint fk\_pagamento\_tipo\_pagamento

foreign key (tipo\_pagamento\_id\_tipo\_pagamento) references tipo\_pagamento (id\_tipo\_pagamento)

);

create index fk\_pagamento\_consulta1\_idx

on pagamento (consulta\_id\_consulta);

create index fk\_pagamento\_tipo\_pagamento\_idx

on pagamento (tipo\_pagamento\_id\_tipo\_pagamento);

**Script que popula as tabelas do Banco de dados**

INSERT INTO paciente (nome\_paciente, cpf\_paciente, telefone, endereco, genero, data\_nasc) VALUES

('Alice Silva', '12345678900', '(11) 1111-1111', 'Rua A, 123', 'F', '1990-01-01'),

('Bruno Souza', '12345678901', '(11) 1111-1112', 'Rua B, 234', 'M', '1991-02-01'),

('Carlos Lima', '12345678902', '(11) 1111-1113', 'Rua C, 345', 'M', '1992-03-01'),

('Daniela Alves', '12345678903', '(11) 1111-1114', 'Rua D, 456', 'F', '1993-04-01'),

('Eduardo Pereira', '12345678904', '(11) 1111-1115', 'Rua E, 567', 'M', '1994-05-01'),

('Fernanda Costa', '12345678905', '(11) 1111-1116', 'Rua F, 678', 'F', '1995-06-01'),

('Gustavo Martins', '12345678906', '(11) 1111-1117', 'Rua G, 789', 'M', '1996-07-01'),

('Helena Rocha', '12345678907', '(11) 1111-1118', 'Rua H, 890', 'F', '1997-08-01'),

('Igor Ribeiro', '12345678908', '(11) 1111-1119', 'Rua I, 901', 'M', '1998-09-01'),

('Juliana Fernandes', '12345678909', '(11) 1111-1120', 'Rua J, 012', 'F', '1999-10-01');

INSERT INTO procedimento (nome\_procedimento, descricao\_procedimento) VALUES

('Limpeza', 'Limpeza dentária completa'),

('Extração', 'Extração de dente'),

('Canal', 'Tratamento de canal'),

('Clareamento', 'Clareamento dental'),

('Implante', 'Implante dentário'),

('Ortodontia', 'Aparelho ortodôntico'),

('Prótese', 'Prótese dentária'),

('Restauração', 'Restauração dentária'),

('Consulta inicial', 'Consulta de avaliação'),

('Raio-X', 'Exame de Raio-X');

INSERT INTO odontologista (nome\_odontologista, especialidade, cro) VALUES

('Dr. Pedro', 'Ortodontia', 'CRO-12345'),

('Dr. Ana', 'Implantes', 'CRO-12346'),

('Dr. Luiz', 'Cirurgia', 'CRO-12347'),

('Dra. Carla', 'Estética', 'CRO-12348'),

('Dr. Marcos', 'Prótese', 'CRO-12349'),

('Dra. Fernanda', 'Clínico Geral', 'CRO-12350'),

('Dr. Ricardo', 'Endodontia', 'CRO-12351'),

('Dra. Julia', 'Pediatria', 'CRO-12352'),

('Dr. Felipe', 'Periodontia', 'CRO-12353'),

('Dra. Sofia', 'Radiologia', 'CRO-12354');

INSERT INTO agendamento (data\_agendamento, status\_agendamento, paciente\_id\_paciente, odontologista\_id\_odontologista) VALUES

('2024-06-15 09:00:00', 'Agendado', 1, 1),

('2024-06-15 10:00:00', 'Agendado', 2, 2),

('2024-06-15 11:00:00', 'Agendado', 3, 3),

('2024-06-15 12:00:00', 'Agendado', 4, 4),

('2024-06-15 13:00:00', 'Agendado', 5, 5),

('2024-06-15 14:00:00', 'Agendado', 6, 6),

('2024-06-15 15:00:00', 'Agendado', 7, 7),

('2024-06-15 16:00:00', 'Agendado', 8, 8),

('2024-06-15 17:00:00', 'Agendado', 9, 9),

('2024-06-15 18:00:00', 'Agendado', 10, 10);

INSERT INTO consulta (data\_hora, diagnostico, tratamento, agendamento\_id\_agendamento) VALUES

('2024-06-15 09:00:00', 'Cárie detectada', 'Tratamento de restauração', 1),

('2024-06-15 10:00:00', 'Necessidade de extração', 'Extração do dente afetado', 2),

('2024-06-15 11:00:00', 'Canal necessário', 'Início do tratamento de canal', 3),

('2024-06-15 12:00:00', 'Manchas nos dentes', 'Clareamento dentário', 4),

('2024-06-15 13:00:00', 'Dente ausente', 'Colocação de implante', 5),

('2024-06-15 14:00:00', 'Aparelho necessário', 'Colocação de aparelho ortodôntico', 6),

('2024-06-15 15:00:00', 'Necessidade de prótese', 'Confecção de prótese dentária', 7),

('2024-06-15 16:00:00', 'Restauração danificada', 'Restauração do dente', 8),

('2024-06-15 17:00:00', 'Primeira consulta', 'Avaliação inicial', 9),

('2024-06-15 18:00:00', 'Checar dente', 'Exame de Raio-X', 10);

INSERT INTO tipo\_pagamento (descricao\_tipo\_pagamento) VALUES

('Cartão de Crédito'),

('Cartão de Débito'),

('Dinheiro'),

('Boleto');

INSERT INTO pagamento (valor\_pago, data\_pagamento, tipo\_pagamento\_id\_tipo\_pagamento, consulta\_id\_consulta) VALUES

(100.00, '2024-06-15 09:30:00', 1, 1),

(200.00, '2024-06-15 10:30:00', 2, 2),

(500.00, '2024-06-15 11:30:00', 3, 3),

(300.00, '2024-06-15 12:30:00', 4, 4),

(1500.00, '2024-06-15 13:30:00', 1, 5),

(1000.00, '2024-06-15 14:30:00', 2, 6),

(1200.00, '2024-06-15 15:30:00', 3, 7),

(250.00, '2024-06-15 16:30:00', 4, 8),

(150.00, '2024-06-15 17:30:00', 1, 9),

(80.00, '2024-06-15 18:30:00', 2, 10);

INSERT INTO consulta\_procedimento (consulta\_id\_consulta, procedimento\_id\_procedimento) VALUES

(1, 8), -- Tratamento de restauração -> Restauração

(2, 2), -- Extração do dente afetado -> Extração

(3, 3), -- Início do tratamento de canal -> Canal

(4, 4), -- Clareamento dentário -> Clareamento

(5, 5), -- Colocação de implante -> Implante

(6, 6), -- Colocação de aparelho ortodôntico -> Ortodontia

(7, 7), -- Confecção de prótese dentária -> Prótese

(8, 8), -- Restauração do dente -> Restauração

(9, 9), -- Avaliação inicial -> Consulta inicial

(10, 10); -- Exame de Raio-X -> Raio-X

**Principais consultas mapeadas baseadas em regras de negócio (mínimo 4) – usando indexes, procedures, functions e triggers.**

-- 1.Qual a especialidade dos ortodontistas com maior quantidade de agendamentos nos últimos 4 meses?

CREATE INDEX idx\_agendamento\_data ON agendamento (data\_agendamento); -- filtrar agendamento pela coluna data\_agendamento

CREATE INDEX idx\_agendamento\_odontologista ON agendamento (odontologista\_id\_odontologista); -- unir agendamento com odontologista

CREATE INDEX idx\_consulta\_procedimento\_consulta ON consulta\_procedimento (consulta\_id\_consulta); -- unir consulta\_procedimento com consulta

CREATE INDEX idx\_consulta\_agendamento ON consulta (agendamento\_id\_agendamento); -- unir consulta e agendamento

CREATE INDEX idx\_odontologista\_especialidade ON odontologista (especialidade); -- necessário para melhorar o desempenho do filtro por especialidade

CREATE INDEX idx\_procedimento\_nome ON procedimento (nome\_procedimento); -- necessário para melhorar o desempenho da agregação por nome\_procedimento

-- Remover o procedimento armazenado existente

DROP PROCEDURE IF EXISTS especialidade\_agendamento\_ultimos\_4\_meses;

-- Criar o procedimento armazenado

CREATE PROCEDURE especialidade\_agendamento\_ultimos\_4\_meses()

BEGIN

DECLARE top\_especialidade VARCHAR(255);

-- Subquery para achar a maior quantidade de especialidades

SELECT

o.especialidade

INTO

top\_especialidade

FROM

agendamento a

JOIN

odontologista o ON a.odontologista\_id\_odontologista = o.id\_odontologista

WHERE

a.data\_agendamento >= DATE\_SUB(CURDATE(), INTERVAL 4 MONTH)

GROUP BY

o.especialidade

ORDER BY

COUNT(a.id\_agendamento) DESC

LIMIT 1;

-- Query principal para achar a maior quantidade de procedimentos para a maior quantidade de especialidades

SELECT

p.nome\_procedimento,

COUNT(cp.procedimento\_id\_procedimento) AS total\_procedimentos

FROM

consulta\_procedimento cp

JOIN

consulta c ON cp.consulta\_id\_consulta = c.id\_consulta

JOIN

agendamento a ON c.agendamento\_id\_agendamento = a.id\_agendamento

JOIN

odontologista o ON a.odontologista\_id\_odontologista = o.id\_odontologista

JOIN

procedimento p ON cp.procedimento\_id\_procedimento = p.id\_procedimento

WHERE

o.especialidade = top\_especialidade

GROUP BY

p.nome\_procedimento

ORDER BY

total\_procedimentos DESC

LIMIT 1;

END

-- Rodar a procedure

CALL especialidade\_agendamento\_ultimos\_4\_meses();

--------------------------------------------------------------------------------------------------------------------------------------

-- 2 - Qual é a melhor abordagem para implementar uma função que verifique eficientemente a existência de um agendamento ativo para um paciente específico, considerando a otimização de desempenho em um banco de dados relacional?

CREATE INDEX idx\_agendamento\_paciente\_status

ON agendamento (paciente\_id\_paciente, status\_agendamento); -- Otimizar queries que buscam agendamentos por paciente ou por status.

CREATE INDEX idx\_pagamento\_consulta\_tipo\_pagamento

ON pagamento (consulta\_id\_consulta, tipo\_pagamento\_id\_tipo\_pagamento); -- Otimizar queries que buscam pagamentos associados a consultas ou tipos de pagamentos específicos.

CREATE FUNCTION verifica\_agendamento\_ativo(

paciente\_id INT

) RETURNS BOOLEAN

BEGIN

DECLARE count\_agendamentos INT;

SELECT COUNT(\*) INTO count\_agendamentos

FROM agendamento

WHERE paciente\_id\_paciente = paciente\_id

AND status\_agendamento = 'Agendado';

IF count\_agendamentos > 0 THEN

RETURN TRUE; -- Já existe um agendamento ativo para o paciente

ELSE

RETURN FALSE; -- Nenhum agendamento ativo para o paciente

END IF;

END

-- Utilizando a função para verificar se há agendamentos ativos para o paciente

SELECT verifica\_agendamento\_ativo(id\_paciente) AS tem\_agendamento\_ativo;

-----------------------------------------------------------------------------------------------------------------------------------------------

-- 3 - Como pode ser implementado um sistema para registrar as alterações de dados na tabela "pagamento", visando fins auditoriais e fiscais, que inclua não apenas os detalhes do pagamento, mas também o nome do usuário do banco responsável pela modificação?

-- Definição dos triggers para utilizar na tabela.

DROP TRIGGER IF EXISTS AfterInsertPagamento;

DROP TRIGGER IF EXISTS AfterUpdatePagamento;

DROP TRIGGER IF EXISTS AfterDeletePagamento;

-- Pra que isso possa ser alcançado, criamos uma tabela LogPagamento.

CREATE TRIGGER AfterInsertPagamento

AFTER INSERT ON pagamento

FOR EACH ROW

BEGIN

INSERT INTO LogPagamento (ActionType, id\_pagamento, consulta\_id\_consulta, tipo\_pagamento\_id\_tipo\_pagamento, valor\_pago, data\_pagamento, ExecutedBy)

VALUES ('INSERT', NEW.id\_pagamento, NEW.consulta\_id\_consulta, NEW.tipo\_pagamento\_id\_tipo\_pagamento, NEW.valor\_pago, NEW.data\_pagamento, USER());

END;

CREATE TRIGGER AfterUpdatePagamento

AFTER UPDATE ON pagamento

FOR EACH ROW

BEGIN

INSERT INTO LogPagamento (ActionType, id\_pagamento, consulta\_id\_consulta, tipo\_pagamento\_id\_tipo\_pagamento, valor\_pago, data\_pagamento, ExecutedBy)

VALUES ('UPDATE', NEW.id\_pagamento, NEW.consulta\_id\_consulta, NEW.tipo\_pagamento\_id\_tipo\_pagamento, NEW.valor\_pago, NEW.data\_pagamento, USER());

END;

CREATE TRIGGER AfterDeletePagamento

AFTER DELETE ON pagamento

FOR EACH ROW

BEGIN

INSERT INTO LogPagamento (ActionType, id\_pagamento, consulta\_id\_consulta, tipo\_pagamento\_id\_tipo\_pagamento, valor\_pago, data\_pagamento, ExecutedBy)

VALUES ('DELETE', OLD.id\_pagamento, OLD.consulta\_id\_consulta, OLD.tipo\_pagamento\_id\_tipo\_pagamento, OLD.valor\_pago, OLD.data\_pagamento, USER());

END;

-----------------------------------------------------------------------------------------------------------------------------

-- 4 - Qual é o meio de pagamento mais utlizado, nos ultimos 6 meses, entre pacientes do sexo feminino na faixa etária de 18 a 55 anos, em ordem decrescente?

DROP IF EXISTS PROCEDURE RelatorioPagamentosPorGeneroIdade

-- Dropa a procedure, se existir.

CREATE PROCEDURE RelatorioPagamentosPorGeneroIdade()

BEGIN

SELECT

tp.descricao\_tipo\_pagamento AS meio\_pagamento,

COUNT(\*) AS quantidade\_uso

FROM

pagamento pg

INNER JOIN consulta c ON pg.consulta\_id\_consulta = c.id\_consulta

INNER JOIN agendamento a ON c.agendamento\_id\_agendamento = a.id\_agendamento

INNER JOIN paciente p ON a.paciente\_id\_paciente = p.id\_paciente

INNER JOIN tipo\_pagamento tp ON pg.tipo\_pagamento\_id\_tipo\_pagamento = tp.id\_tipo\_pagamento

WHERE

p.genero = 'F'

AND TIMESTAMPDIFF(YEAR, p.data\_nasc, CURDATE()) BETWEEN 18 AND 55

AND pg.data\_pagamento >= DATE\_SUB(CURDATE(), INTERVAL 6 MONTH)

GROUP BY

tp.descricao\_tipo\_pagamento

ORDER BY

quantidade\_uso DESC;

END

CALL RelatorioPagamentosPorGeneroIdade();

-----------------------------------------------------------------------------------------

-- 5 - Qual o diagnóstico que teve a maior incidência em pacientes com idades superiores à 16 anos em ambos os gêneros durante os 6 primeiros meses do ano?

CREATE INDEX idx\_paciente\_data\_nasc ON paciente (data\_nasc); -- Melhora a performance de consultas que filtram ou ordenam pacientes por data de nascimento.

CREATE INDEX idx\_consulta\_data\_hora ON consulta (data\_hora); -- Útil para queries que filtram ou ordenam consultas por data e hora.

CREATE INDEX idx\_agendamento\_paciente ON agendamento (paciente\_id\_paciente); -- Útil para queries que precisam localizar agendamentos associados a um paciente específico.

DROP PROCEDURE IF EXISTS diagnostico\_maior\_incidencia\_acima\_16\_anos -- Dropar a procedure, se já existir.

-- Dropa a procedure, se existir.

CREATE PROCEDURE diagnostico\_maior\_incidencia\_acima\_16\_anos()

BEGIN

-- Variável para armazenar o diagnóstico com maior incidência

DECLARE top\_diagnostico VARCHAR(255);

-- Selecionar o diagnóstico com maior incidência nos primeiros seis meses do ano

SELECT

c.diagnostico

INTO

top\_diagnostico

FROM

consulta c

JOIN

agendamento a ON c.agendamento\_id\_agendamento = a.id\_agendamento

JOIN

paciente p ON a.paciente\_id\_paciente = p.id\_paciente

WHERE

TIMESTAMPDIFF(YEAR, p.data\_nasc, CURDATE()) > 16

AND c.data\_hora BETWEEN '2024-01-01' AND '2024-06-30'

GROUP BY

c.diagnostico

ORDER BY

COUNT(\*) DESC

LIMIT 1;

-- Exibir o resultado

SELECT top\_diagnostico AS Diagnostico\_Maior\_Incidencia\_Acima\_16\_Anos;

END

CALL diagnostico\_maior\_incidencia\_acima\_16\_anos();