UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI FACULDADE DE CIÊNCIAS EXATAS DEPARTAMENTO DE COMPUTAÇÃO

Ian Patrick Campos Lima
Maria Aparecida Gomes Vieira
Matheus Henrique de Paula
Miguel Moreira Costa
Yago Christian Santos Brito

TRABALHO DA DISCIPLINA DE BANCO DE DADOS I

ESPECIFICAÇÃO CONCEITUAL, LÓGICA E IMPLEMENTAÇÃO DE UMA BASE DE DADOS RELACIONAL

Diamantina

2025

1 – APRESENTAÇÃO DO PROBLEMA

Corretora de Seguros

A Não é Culpa Nossa Seguros é uma corretora especializada na comercialização e gerenciamento de seguros de vida, automóveis e residenciais. A empresa possui uma base diversificada de clientes e atua com apólices de diferentes valores e coberturas. Cada seguro é identificado por um código único e contém informações como tipo (vida, automóvel, residencial), nome comercial, valor mensal e descrição da cobertura.

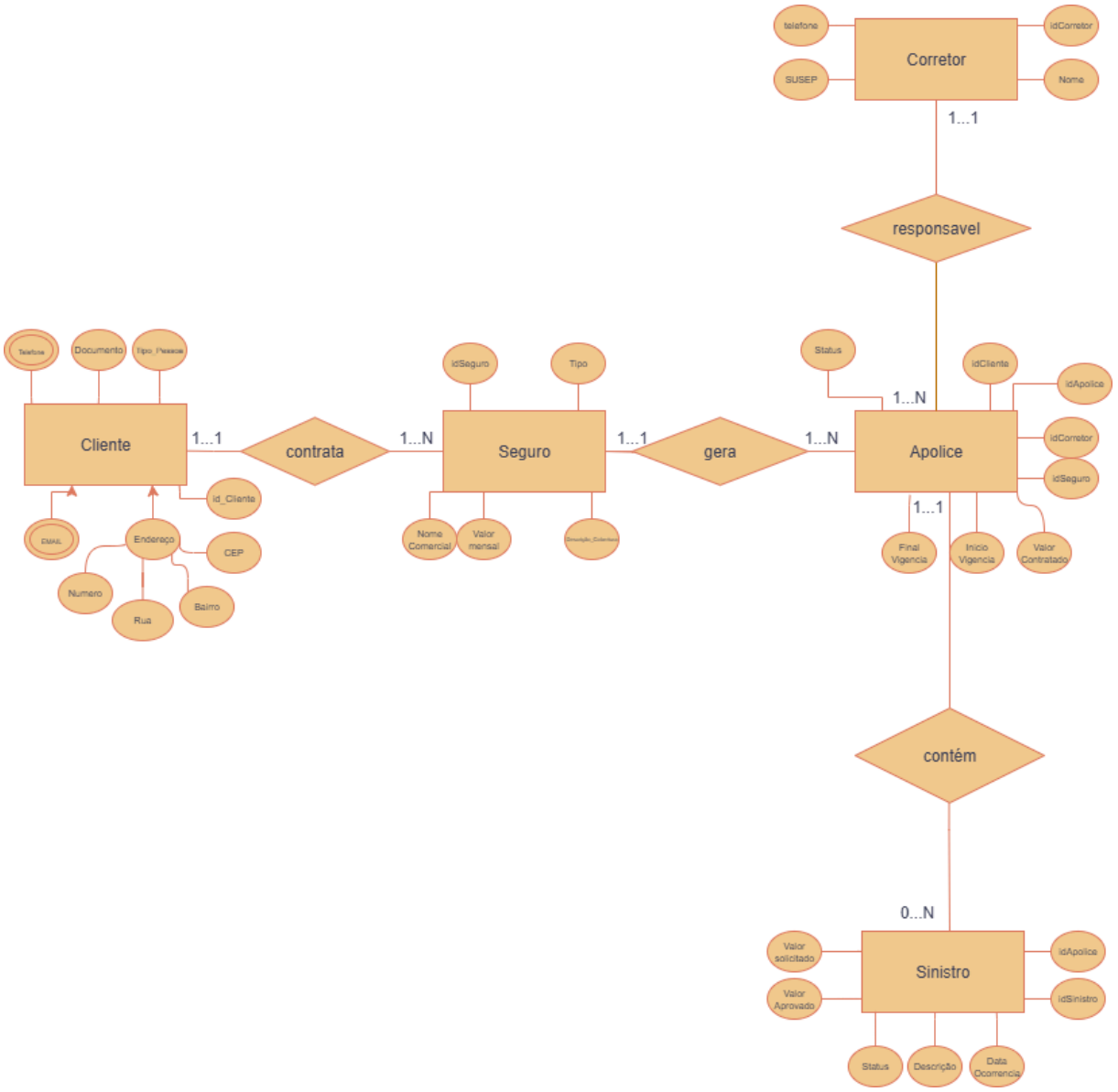
Os clientes são cadastrados com número de registro único, nome completo, CPF, endereço completo (rua, bairro, número e cidade), além de telefones e e-mails (aceitando múltiplos de cada). Cada cliente pode contratar um ou mais seguros, e cada contrato é registrado como uma apólice, que armazena a data de início, data de término, valor contratado e status (ativo, encerrado, cancelado).

A empresa conta com uma equipe de corretores, cada um com matrícula, nome, telefone e registro SUSEP, responsável por intermediar as vendas das apólices. Um corretor pode ser responsável por várias apólices, e cada apólice pode estar vinculada a apenas um corretor.

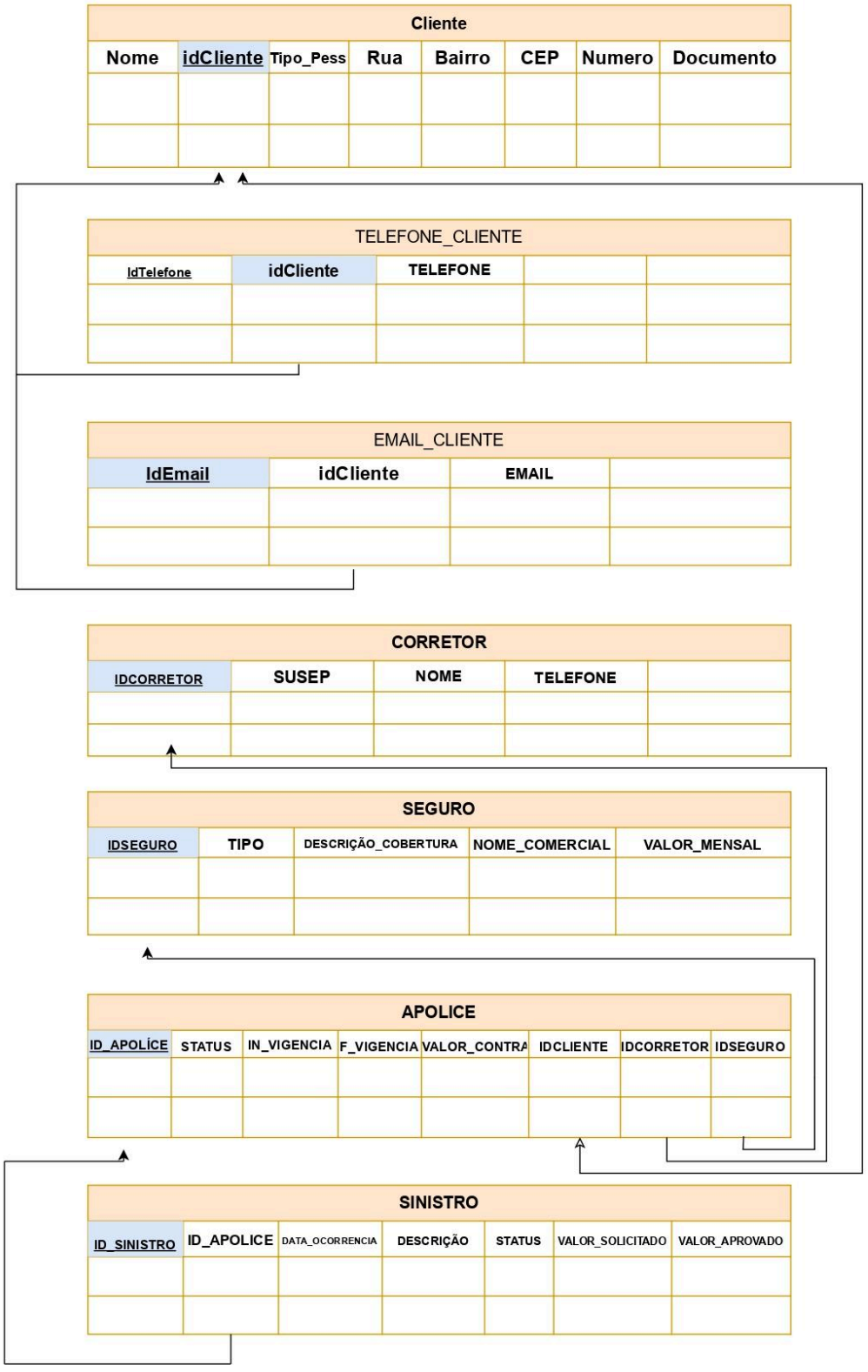
Em casos de ocorrência, o cliente pode registrar um sinistro, vinculado à apólice, que possui um número único, data da ocorrência, descrição, status (em análise, aprovado, negado), valor solicitado e valor aprovado.

O sistema precisa fornecer recursos para cadastro e controle de clientes, apólices, corretores e sinistros, além de oferecer relatórios financeiros que demonstrem o total arrecadado por mês, valor de sinistros pagos e nível de inadimplência dos clientes.

2 – MODELAGEM CONCEITUAL



3 – MODELO LÓGICO DA BASE DE DADOS



4– SCRIPTS DE IMPLEMENTAÇÃO DA BASE DE DADOS

4.1 – Comandos create table

Um comando create table, em SQL, é responsável por criar uma nova tabela em um banco de dados. Em nosso trabalho, utilizamos o comando da seguinte forma:

```
-- Criação do banco de dados

CREATE DATABASE CorretoraNCN;

USE CorretoraNCN;


-- Tabela Cliente

CREATE TABLE Cliente (

    idCliente INT PRIMARY KEY AUTO_INCREMENT,

    Nome VARCHAR(100) NOT NULL,

    Tipo_Pess VARCHAR(20) NOT NULL,

    Rua VARCHAR(100),

    Bairro VARCHAR(100),

    CEP VARCHAR(10),

    Numero VARCHAR(10),

    Documento VARCHAR(20) UNIQUE -- CPF ou CNPJ único

);


-- Tabela Telefone_Cliente

CREATE TABLE Telefone_Cliente (

    IdTelefone INT PRIMARY KEY AUTO_INCREMENT,

    idCliente INT NOT NULL,

    Telefone VARCHAR(15) NOT NULL UNIQUE, -- impede duplicidade de telefones
```

```
    FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente)
);

-- Tabela Email_Cliente
CREATE TABLE Email_Cliente (
    IdEmail INT PRIMARY KEY AUTO_INCREMENT,
    idCliente INT NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE, -- impede duplicidade de e-mails
    FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente)
);

-- Tabela Corretor
CREATE TABLE Corretor (
    idCorretor INT PRIMARY KEY AUTO_INCREMENT,
    SUSEP VARCHAR(20) UNIQUE, -- registro único
    Nome VARCHAR(100) NOT NULL,
    Telefone VARCHAR(15) UNIQUE -- impede telefones duplicados entre corretores
);

-- Tabela Seguro
CREATE TABLE Seguro (
    idSeguro INT PRIMARY KEY AUTO_INCREMENT,
    Tipo VARCHAR(50) NOT NULL,
    Descricao_Cobertura VARCHAR(200),
    Nome_Comercial VARCHAR(100),
    Valor_Mensal DECIMAL(10,2) CHECK (Valor_Mensal >= 0)
```

```
);

-- Tabela Apolice
CREATE TABLE Apolice (
    id_Apolice INT PRIMARY KEY AUTO_INCREMENT,
    Status VARCHAR(20) NOT NULL,
    In_Vigencia DATE,
    F_Vigencia DATE,
    Valor_Contrato DECIMAL(10,2) CHECK (Valor_Contrato >= 0),
    idCliente INT NOT NULL,
    idCorretor INT NOT NULL,
    idSeguro INT NOT NULL,
    FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente),
    FOREIGN KEY (idCorretor) REFERENCES Corretor(idCorretor),
    FOREIGN KEY (idSeguro) REFERENCES Seguro(idSeguro)
);

-- Tabela Sinistro
CREATE TABLE Sinistro (
    id_Sinistro INT PRIMARY KEY AUTO_INCREMENT,
    id_Apolice INT NOT NULL,
    Data_Ocorrencia DATE,
    Descricao VARCHAR(200),
    Status VARCHAR(20),
    Valor_Solicitado DECIMAL(10,2) CHECK (Valor_Solicitado >= 0),
    Valor_Aprovado DECIMAL(10,2) CHECK (Valor_Aprovado >= 0),
```

```
    FOREIGN KEY (id_Apolice) REFERENCES Apolice(id_Apolice)
);
```

4.2 – Comandos insert

Um comando insert, em SQL, é responsável por adicionar novos registros a uma tabela de banco de dados. Em nosso trabalho, utilizamos o comando da seguinte forma:

```
INSERT INTO Cliente (Nome, Tipo_Pess, Rua, Bairro, CEP, Numero, Documento)
VALUES
('Ana Pereira', 'Física', 'Rua Flores', 'Centro', '39100-000', '101', '11111111111'),
('Bruno Lima', 'Física', 'Av. Brasil', 'Jardins', '39100-001', '202', '22222222222'),
('Carlos Dias', 'Física', 'Rua São Paulo', 'Centro', '39100-002', '303', '33333333333'),
('Daniela Rocha', 'Física', 'Rua Minas', 'Industrial', '39100-003', '404', '44444444444'),
('Empresa Alfa Ltda', 'Jurídica', 'Av. Getúlio Vargas', 'Comercial', '39100-004', '505', '55555555000155'),
('Fernanda Torres', 'Física', 'Rua Goiás', 'Bela Vista', '39100-005', '606', '66666666666'),
('Gustavo Ramos', 'Física', 'Av. JK', 'Jardim das Palmeiras', '39100-006', '707', '77777777777'),
('Helena Martins', 'Física', 'Rua Tocantins', 'São Pedro', '39100-007', '808', '88888888888'),
('Igor Silva', 'Física', 'Rua Paraná', 'Vila Nova', '39100-008', '909', '99999999999'),
('Joana Costa', 'Física', 'Rua Bahia', 'Santa Rita', '39100-009', '110', '10101010101');
```

```
INSERT INTO Telefone_Cliente (idCliente, Telefone)
VALUES
(1, '38991111111'),
(2, '38992222222'),
```

(3, '38993333333'),

(4, '38994444444'),

(5, '38995555555'),

(6, '38996666666'),

(7, '38997777777'),

(8, '38998888888'),

(9, '38999999999'),

(10, '38991010101');

—----------

INSERT INTO Email_Cliente (idCliente, Email)

VALUES

(1, 'ana.pereira@email.com'),

(2, 'bruno.lima@email.com'),

(3, 'carlos.dias@email.com'),

(4, 'daniela.rocha@email.com'),

(5, 'contato@empresaalfa.com.br'),

(6, 'fernanda.torres@email.com'),

(7, 'gustavo.ramos@email.com'),

(8, 'helena.martins@email.com'),

(9, 'igor.silva@email.com'),

(10, 'joana.costa@email.com');

—----------

INSERT INTO Corretor (SUSEP, Nome, Telefone)

VALUES

('SUSEP001', 'Ricardo Almeida', '38990000001'),

('SUSEP002', 'Luciana Freitas', '38990000002');

INSERT INTO Seguro (Tipo, Descricao_Cobertura, Nome_Comercial, Valor_Mensal)

VALUES

('Automóvel', 'Cobertura contra colisão e furto', 'AutoSeguro+', 150.00),

('Residencial', 'Cobertura contra incêndio e roubo', 'CasaSegura', 100.00);

—----------

INSERT INTO Apolice (Status, In_Vigencia, F_Vigencia, Valor_Contrato, idCliente, idCorretor, idSeguro)

VALUES

('Ativa', '2025-01-01', '2025-12-31', 1200.00, 1, 1, 1),

('Ativa', '2025-02-01', '2026-01-31', 1000.00, 2, 2, 2),

('Cancelada', '2024-01-01', '2024-12-31', 1500.00, 3, 1, 1),

('Ativa', '2025-03-01', '2026-02-28', 1300.00, 4, 2, 2),

('Ativa', '2025-04-01', '2026-03-31', 1800.00, 5, 1, 1),

('Ativa', '2025-05-01', '2026-04-30', 1100.00, 6, 2, 2),

('Ativa', '2025-06-01', '2026-05-31', 1400.00, 7, 1, 1),

('Ativa', '2025-07-01', '2026-06-30', 1600.00, 8, 2, 2),

('Ativa', '2025-08-01', '2026-07-31', 1700.00, 9, 1, 1),

('Ativa', '2025-09-01', '2026-08-31', 1900.00, 10, 2, 2);

—----------

```
INSERT INTO Sinistro (id_Apolice, Data_Ocorrencia, Descricao, Status, Valor_Solicitado, Valor_Aprovado)

VALUES

(1, '2025-02-15', 'Colisão leve traseira', 'Aprovado', 1200.00, 900.00),

(2, '2025-03-22', 'Incêndio na cozinha', 'Aprovado', 3000.00, 2500.00),

(3, '2024-05-10', 'Roubo de veículo', 'Negado', 20000.00, 0.00),

(5, '2025-06-01', 'Danos por enchente', 'Em análise', 5000.00, 0.00),

(9, '2025-10-11', 'Vandalismo na residência', 'Aprovado', 1500.00, 1000.00);
```

4.3 – Comandos select

Um comando select, em SQL, é responsável por recuperar dados de uma ou mais tabelas. Ele permite consultar e extrair informações específicas com base em critérios definidos. Em nosso trabalho, utilizamos o comando da seguinte forma:

```
SELECT

    c.Nome AS Cliente,

    c.Documento,

    e.Email,

    t.Telefone,

    r.Nome AS Corretor

FROM Cliente c

LEFT JOIN Email_Cliente e ON c.idCliente = e.idCliente

LEFT JOIN Telefone_Cliente t ON c.idCliente = t.idCliente

LEFT JOIN Apolice a ON c.idCliente = a.idCliente

LEFT JOIN Corretor r ON a.idCorretor = r.idCorretor;
```

```
SELECT Nome, Documento FROM Cliente;
```

```
SELECT * FROM Apolice

WHERE Valor_Contrato > 1000;
```