

# CHALLENGE 2024

Plusoft - Mastering Relational and Non-Relational Database

## **Integrantes:**

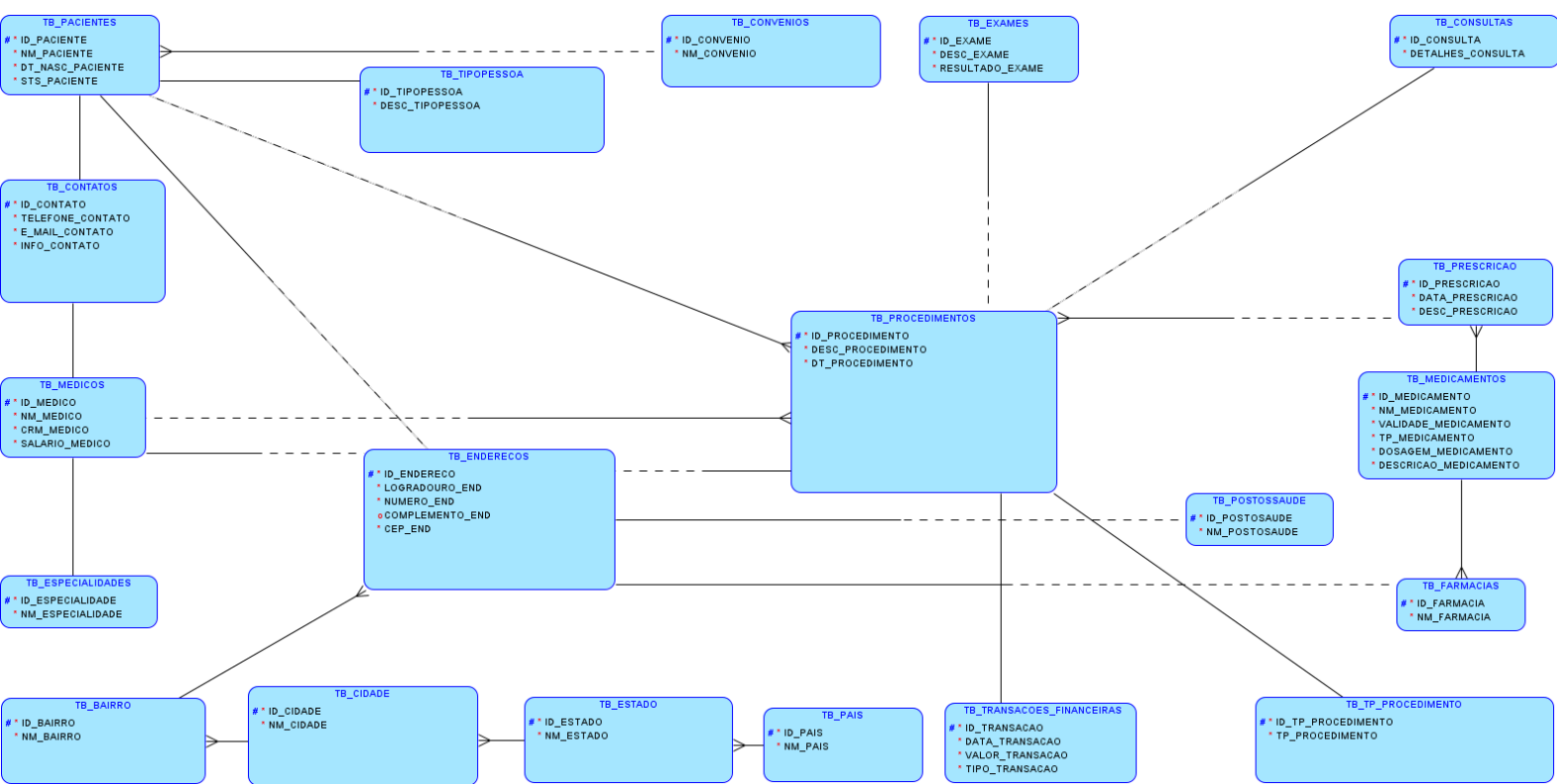
**RM551261 - Giovanni Sguizzardi**

**RM98057 - Nicolas E. Inohue**

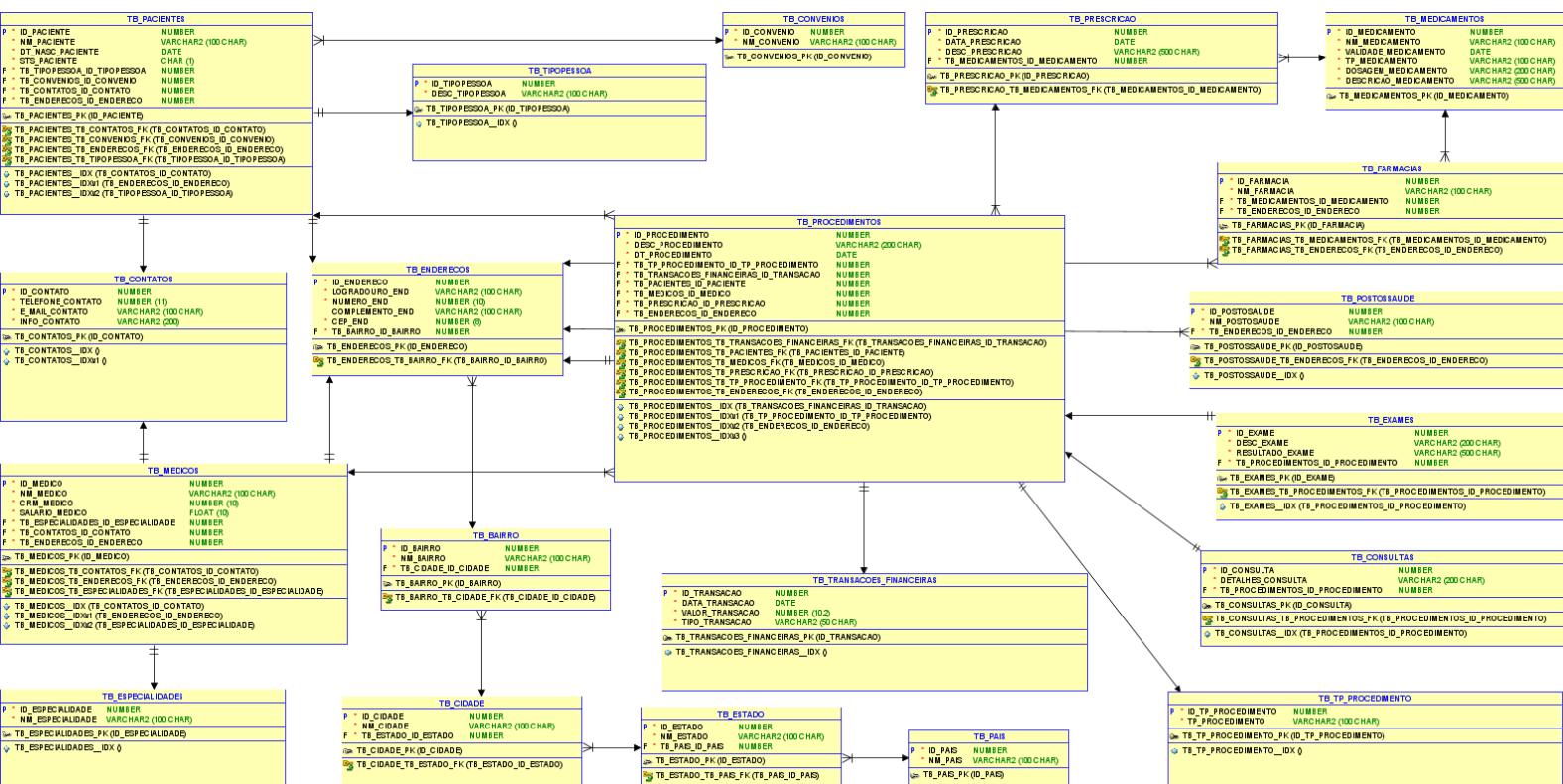
**RM99841 - Marcel Prado Soddano**

**RM99577 - Guilherme Dias Gomes**

# Modelo Logico:



# Modelo Relacional:



O projeto de banco de dados proposto tem como objetivo desenvolver um sistema de gerenciamento de informações para uma clínica médica. O sistema será responsável por armazenar e gerenciar dados relacionados aos médicos e aos medicamentos prescritos aos pacientes. O banco de dados será projetado para facilitar o agendamento de consultas, o registro de informações médicas dos pacientes e o acompanhamento das prescrições de medicamentos.

Além disso, o sistema permitirá que os médicos prescrevam medicamentos aos pacientes. Para isso, será necessário estabelecer uma relação entre as entidades de médicos e medicamentos, permitindo que seja registrado qual médico prescreveu determinado medicamento.

Espera-se que o sistema contribua para uma gestão eficiente das informações médicas da clínica, garantindo precisão, segurança e acessibilidade aos dados.

## Prints das execuções dos scripts:

Validar data de nascimento de um Paciente:

Execução:

```
SQL> SET SERVEROUTPUT ON;
SQL> --Validar data de nascimento de um Paciente:
SQL> CREATE OR REPLACE FUNCTION valida_data_nascimento(p_data_nasc DATE)
2 RETURN VARCHAR2 IS
3   v_data_atual DATE;
4   v_mensagem VARCHAR2(100);
5 BEGIN
6   SELECT SYSDATE INTO v_data_atual FROM dual;
7
8   IF p_data_nasc > v_data_atual THEN
9     v_mensagem := 'Data de nascimento não pode ser no futuro';
10  ELSIF p_data_nasc IS NULL THEN
11    v_mensagem := 'Data de nascimento não pode ser nula';
12  ELSE
13    v_mensagem := 'Data de nascimento válida';
14  END IF;
15
16  RETURN v_mensagem;
17 END;
18 /

FUNCTION created.

Commit complete.
```

Resultado:

```
SQL> SELECT
2  ID_PACIENTE,
3  NM_PACIENTE,
4  valida_data_nascimento_paciente(DT_NASC_PACIENTE) AS status_data_nasc
5 FROM
6  TB_PACIENTES;
```

	ID_PACIENTE	NM_PACIENTE	STATUS_DATA_NASC
<input type="checkbox"/>		1 Maria Silva	Data de nascimento válida
<input type="checkbox"/>		2 João Santos	Data de nascimento válida
<input type="checkbox"/>		3 Ana Oliveira	Data de nascimento válida
<input type="checkbox"/>		4 Pedro Rocha	Data de nascimento válida
<input type="checkbox"/>		5 Carla Souza	Data de nascimento válida

Page 1 of 1 | 1-5 of 5 rows

Validar CEP de um endereço:

Execução:


```
SQL> SET SERVEROUTPUT ON;
SQL> CREATE OR REPLACE FUNCTION valida_cep(p_cep VARCHAR2)
2  RETURN VARCHAR2 IS
3    v_mensagem VARCHAR2(100);
4  BEGIN
5    IF p_cep IS NULL THEN
6      v_mensagem := 'CEP não pode ser nulo';
7
8    ELSIF LENGTH(p_cep) != 8 OR NOT REGEXP_LIKE(p_cep, '^[0-9]{8}$') THEN
9      v_mensagem := 'CEP inválido. Deve conter exatamente 8 dígitos numéricos';
10
11    ELSE
12      v_mensagem := 'CEP válido';
13    END IF;
14
15    RETURN v_mensagem;
16 END;
17 /


FUNCTION created.


Commit complete.
```

Resultado:


```
SQL> SELECT
2  ID_ENDERECO,
3  LOGRADOURO_END,
4  valida_cep(CEP_END) AS status_cep
5 FROM
6  TB_ENDERECOS;
```


 Select all rows















 Save as:

JSON







	ID_ENDERECO	LOGRADOURO_END	STATUS_CEP
		1 Rua das Flores	CEP válido
		2 Rua das Flores	CEP válido
		3 Rua dos Jardins	CEP válido
		4 Avenida dos Navegantes	CEP válido
		5 Rua das Palmeiras	CEP válido
		6 Rua dos Lírios	CEP válido
		7 Rua dos Pinheiros	CEP válido
		8 Avenida Atlântica	CEP válido
		9 Rua 24 de Outubro	CEP válido
		10 Rua do Imperador	CEP válido

PROCEDURE 'gerencia\_paciente' INSERT/UPDATE/DELETE:

Execução:

```

SQL> SET SERVEROUTPUT ON;
SQL> CREATE OR REPLACE PROCEDURE gerencia_paciente(
2   p_acao IN VARCHAR2,
3   id_in IN NUMBER,
4   nome_in IN VARCHAR2 DEFAULT NULL,
5   dt_nasc_in IN DATE DEFAULT NULL,
6   sts_paciente_in IN CHAR DEFAULT NULL,
7   tp_pessoa_in IN NUMBER DEFAULT NULL,
8   convenio_in IN NUMBER DEFAULT NULL,
9   contato_in IN NUMBER DEFAULT NULL,
10  endereco_in IN NUMBER DEFAULT NULL
11 ) AS
12 BEGIN
13   IF p_acao = 'INSERT' THEN
14     INSERT INTO tb_pacientes(
15       ID_PACIENTE, NM_PACIENTE, DT_NASC_PACIENTE, STS_PACIENTE,
16       TB_TIPOPESSOA_ID_TIPOPESSOA, TB_CONVENIOS_ID_CONVENIO,
17       TB_CONTATOS_ID_CONTATO, TB_ENDERECOS_ID_ENDERECO
18     ) VALUES (
19       id_in, nome_in, dt_nasc_in, sts_paciente_in, tp_pessoa_in,
20       convenio_in, contato_in, endereco_in
21     );
22     DBMS_OUTPUT.PUT_LINE('Paciente inserido com sucesso.');
```

```

23   ELSIF p_acao = 'UPDATE' THEN
24     UPDATE tb_pacientes
25     SET NM_PACIENTE = nome_in,
26         DT_NASC_PACIENTE = dt_nasc_in,
27         STS_PACIENTE = sts_paciente_in,
28         TB_TIPOPESSOA_ID_TIPOPESSOA = tp_pessoa_in,
29         TB_CONVENIOS_ID_CONVENIO = convenio_in,
30         TB_CONTATOS_ID_CONTATO = contato_in,
31         TB_ENDERECOS_ID_ENDERECO = endereco_in
32     WHERE ID_PACIENTE = id_in;
33     DBMS_OUTPUT.PUT_LINE('Paciente atualizado com sucesso.');
```

```

34   ELSIF p_acao = 'DELETE' THEN
35     DELETE FROM tb_pacientes WHERE ID_PACIENTE = id_in;
36     DBMS_OUTPUT.PUT_LINE('Paciente excluído com sucesso.');
```

```

37   ELSE
38     DBMS_OUTPUT.PUT_LINE('Ação inválida. Use "INSERT", "UPDATE" ou "DELETE".');
```

```

39   END IF;
40 END;
41 /

```

PROCEDURE created.

Commit complete.

---

PROCEDURE COM INNER JOIN GET\_NOME\_PACIENTE\_BY\_ID:

Execução:



```

SQL> CREATE OR REPLACE PROCEDURE get_nome_paciente_by_id(p_id_paciente IN NUMBER) AS
2  CURSOR c_paciente IS
3  SELECT
4      p.NM_PACIENTE,
5      c.NM_CONVENIO
6  FROM
7      TB_PACIENTES p
8  JOIN
9      TB_CONVENIOS c ON p.TB_CONVENIOS_ID_CONVENIO = c.ID_CONVENIO
10 WHERE
11     p.ID_PACIENTE = p_id_paciente;
12
13 v_paciente_name TB_PACIENTES.NM_PACIENTE%TYPE;
14 v_convenio_name TB_CONVENIOS.NM_CONVENIO%TYPE;
15 BEGIN
16     OPEN c_paciente;
17
18     LOOP
19         FETCH c_paciente INTO v_paciente_name, v_convenio_name;
20
21         EXIT WHEN c_paciente%NOTFOUND;
22
23         DBMS_OUTPUT.PUT_LINE('Nome do paciente com ID ' || p_id_paciente || ' : ' || v_paciente_name);
24         DBMS_OUTPUT.PUT_LINE('Nome do convênio: ' || v_convenio_name);
25     END LOOP;
26
27     CLOSE c_paciente;
28
29     IF c_paciente%ROWCOUNT = 0 THEN
30         DBMS_OUTPUT.PUT_LINE('Paciente com ID ' || p_id_paciente || ' não encontrado.');

```

PROCEDURE created.

Commit complete.

Resultado:

```
SQL> BEGIN
2  get_nome_paciente_by_id(1);
3  END;
4  /
```

Nome do paciente com ID 1: Maria Silva  
Nome do convênio: Unimed

PL/SQL procedure successfully completed.  
Commit complete.

---

#### PROCEDURES COM REGRA DE NEGOCIO:

-INSERIR ENDERECO COM CEP VALIDO:

Execução:

```

SQL> CREATE OR REPLACE PROCEDURE inserir_endereco(
2   p_id_endereco IN NUMBER,
3   p_logradouro_end IN VARCHAR2,
4   p_numero_end IN VARCHAR2,
5   p_complemento_end IN VARCHAR2,
6   p_cep_end IN VARCHAR2,
7   p_bairro_id_bairro IN NUMBER
8 )
9 AS
10   v_mensagem_cep VARCHAR2(100);
11 BEGIN
12   -- Chama a função para validar o CEP
13   v_mensagem_cep := valida_cep(p_cep_end);
14
15   -- Verifica se o CEP é válido
16   IF v_mensagem_cep != 'CEP válido' THEN
17     RAISE_APPLICATION_ERROR(-20001, v_mensagem_cep);
18   ELSE
19     -- Insere o registro na tabela de endereços
20     INSERT INTO TB_ENDERECOS (ID_ENDERECO, LOGRADOURO_END, NUMERO_END, COMPLEMENTO_END, CEP_END, TB_BAIRRO_ID_BAIRRO)
21     VALUES (p_id_endereco, p_logradouro_end, p_numero_end, p_complemento_end, p_cep_end, p_bairro_id_bairro);
22
23     -- Confirma a transação
24     COMMIT;
25     DBMS_OUTPUT.PUT_LINE('Endereço inserido com sucesso.');
```

PROCEDURE created.

Commit complete.

Resultado:

```
SQL> BEGIN
2   inserir_endereco(
3       p_id_endereco => 26,
4       p_logradouro_end => 'Rua das Laranjeiras e figueiras',
5       p_numero_end => '1223',
6       p_complemento_end => 'Apto 202',
7       p_cep_end => '87654321',
8       p_bairro_id_bairro => 1
9   );
10 EXCEPTION
11     WHEN OTHERS THEN
12         DBMS_OUTPUT.PUT_LINE('Erro: ' || SQLERRM);
13 END;
14 /
```

Endereço inserido com sucesso.

PL/SQL procedure successfully completed.

Commit complete.

-INSERIR MEDICAMENTO COM DENTRO DA VALIDADE:

Execução:

```

SQL> CREATE OR REPLACE PROCEDURE inserir_medicamento(
2   p_id NUMBER,
3   p_nome IN VARCHAR2,
4   p_data_validade IN DATE,
5   p_tipo IN VARCHAR2,
6   p_dosagem IN VARCHAR2,
7   p_desc IN VARCHAR2
8 )
9 AS
10   v_data_atual DATE := SYSDATE;
11 BEGIN
12   IF p_data_validade < v_data_atual THEN
13     RAISE_APPLICATION_ERROR(-20001, 'O medicamento está fora da validade.');
```

```

14   ELSE
15     INSERT INTO tb_medicamentos (
16       id_medicamento,
17       nm_medicamento,
18       validade_medicamento,
19       tp_medicamento,
20       dosagem_medicamento,
21       descricao_medicamento
22     )
23     VALUES (
24       p_id,
25       p_nome,
26       p_data_validade,
27       p_tipo,
28       p_dosagem,
29       p_desc
30     );
31     COMMIT;
32   END IF;
33 EXCEPTION
34   WHEN OTHERS THEN
35     ROLLBACK;
36     RAISE_APPLICATION_ERROR(-20002, 'Erro ao inserir medicamento: ' || SQLERRM);
37 END;
38 /

```

PROCEDURE created.

Commit complete.

Resultado:

```
SQL> BEGIN
2   inserir_medicamento(
3     p_id => 6,
4     p_nome => 'Paracetamol',
5     p_data_validade => TO_DATE('2025-12-31', 'YYYY-MM-DD'),
6     p_tipo => 'Analgesico',
7     p_dosagem => '500mg',
8     p_desc => 'Usado para aliviar dor e febre'
9   );
10   DBMS_OUTPUT.PUT_LINE('Medicamento inserido com sucesso. ');
11 EXCEPTION
12   WHEN OTHERS THEN
13     DBMS_OUTPUT.PUT_LINE('Erro: ' || SQLERRM);
14 END;
15 /
```

Medicamento inserido com sucesso.

PL/SQL procedure successfully completed.

Commit complete.

