**Mastering Relational and Non-Relational Database**

Paulo André Carminati – RM557881 2TDSPZ  
Arthur Bispo de Lima – RM 557568 2TDSPV  
João Paulo Moreira - RM557808 2TDSPV

Link do vídeo aplicação java:youtube&logoColor=white" alt="Vídeo de Apresentação"> </a> </p>

**Bem-vindo ao GS Alerta Desastres!** Uma aplicação full-stack robusta e interativa, projetada para monitorar eventos de desastres naturais em tempo real, fornecer informações cruciais e permitir o disparo de alertas para usuários cadastrados. Este projeto foi desenvolvido com dedicação pela equipe **MetaMind** para a Global Solution 2025 (1º Semestre) da FIAP.

# Índice

* Mastering Relational and Non-Relational Database
* Índice
* [[1]](#footnote-1). Visão Geral 🌍
* 2. Estrutura do Banco de Dados 🏗
  1. Tabelas Principais
  2. Tabelas de Junção (Relacionamento)
  3. Sequences
  4. Triggers
  5. Constraints e Chaves
* 3. Funcionalidades Implementadas (PL/SQL) ⚙
  1. Procedures de Inserção, Atualização e Exclusão
  2. Funções de Processamento de Dados
* 4. Consultas Complexas (Relatórios SQL) 📊
* 5. Blocos Anônimos e Cursores 💻
* 6. Link do Repositório
* 7. Tecnologias Utilizadas

# Estrutura do Banco de Dados 🏗

O esquema é composto por tabelas principais para armazenar entidades distintas, tabelas de junção para gerenciar relacionamentos muitos-para-muitos, sequences para geração automática de IDs e triggers para automatizar a inserção de chaves primárias.

## Tabelas Principais

* **TB\_CLIENTE3** : Armazena os dados cadastrais dos clientes.

**Colunas:** ID\_CLIENTE (PK), NOME , SOBRENOME , DOCUMENTO , DATA\_NASCIMENTO .

* **TB\_CONTATO3** : Mantém os diversos tipos de contato associados aos clientes.

**Colunas:** ID\_CONTATO (PK), DDD , TELEFONE , CELULAR , WHATSAPP , EMAIL , TIPO\_CONTATO .

* 🏠 **TB\_ENDERECO3** : Contém as informações detalhadas dos endereços.

**Colunas:** ID\_ENDERECO (PK), CEP , LOGRADOURO , NUMERO , BAIRRO , LOCALIDADE , UF , COMPLEMENTO , LATITUDE , LONGITUDE .

* ☄ **TB\_EONET3** : Armazena dados sobre eventos naturais obtidos do EONET.

**Colunas:** ID\_EONET (PK), EONET\_ID (ID externo), DATA , JSON (detalhes do evento).

## Tabelas de Junção (Relacionamento)

* **TB\_CLIENTECONTATO3** : Associa clientes a seus múltiplos contatos (N:M).
* **TB\_CLIENTEENDERECO3** : Associa clientes a seus múltiplos endereços (N:M).
* **TB\_ENDERECOEVENTOS3** : Associa endereços a eventos EONET (N:M).

## Sequences

Quatro sequences são utilizadas para gerar automaticamente os valores das chaves primárias, garantindo IDs únicos: - TB\_CLIENTE3\_ID\_CLIENTE\_SEQ - TB\_CONTATO3\_ID\_CONTATO\_SEQ TB\_ENDERECO3\_ID\_ENDERECO\_SEQ - TB\_EONET3\_ID\_EONET\_SEQ

## Triggers

⚡ Para cada tabela principal, um trigger BEFORE INSERT utiliza a sequence correspondente para popular a coluna de ID automaticamente. - TB\_CLIENTE3\_ID\_CLIENTE\_TRG -

TB\_CONTATO3\_ID\_CONTATO\_TRG - TB\_ENDERECO3\_ID\_ENDERECO\_TRG -

TB\_EONET3\_ID\_EONET\_TRG

## Constraints e Chaves

* **Chaves Primárias (PK):** Cada tabela possui uma chave primária para garantir a unicidade.
* ⛓ **Chaves Estrangeiras (FK):** Definem e mantêm a integridade referencial entre as tabelas.
* **NOT NULL :** Colunas essenciais são definidas como NOT NULL para garantir que sempre contenham dados.

# Funcionalidades Implementadas (PL/SQL) ⚙

Para cada tabela, foram criadas procedures e funções para encapsular as operações de DML, promovendo reusabilidade e segurança.

## Procedures de Inserção, Atualização e Exclusão

Uma API completa de procedures foi criada para gerenciar o ciclo de vida dos dados:

* **P\_...\_INSERT** : Adicionam novos registros. Ex: P\_CLIENTE3\_INSERT .
* **P\_...\_UPDATE** : Modificam dados de registros existentes. Ex: P\_CLIENTE3\_UPDATE .
* **P\_...\_DELETE** : Removem registros e suas associações para manter a integridade. Ex: P\_CLIENTE3\_DELETE .

<details> <summary>📋 Exemplo de Procedure de Inserção</summary>

--

Insere um novo cliente e retorna o ID gerado.

CREATE

OR

REPLACE

PROCEDURE

P\_CLIENTE3\_INSERT

(

p\_nome

IN

TB\_CLIENTE3.NOME%

TYPE

,

p\_sobrenome

IN

TB\_CLIENTE3.SOBRENOME%

TYPE

,

p\_documento

IN

TB\_CLIENTE3.DOCUMENTO%

TYPE

,

p\_data\_nascimento

IN

TB\_CLIENTE3.DATA\_NASCIMENTO%

TYPE

,

p\_new\_id

OUT

TB\_CLIENTE3.ID\_CLIENTE%

TYPE

)

IS

BEGIN

INSERT

INTO

TB\_CLIENTE3

(

NOME,

SOBRENOME,

DOCUMENTO,

DATA\_NASCIMENTO)

VALUES

(

p\_nome,

p\_sobrenome,

p\_documento,

p\_data\_nascimento)

RETURNING

ID\_CLIENTE

INTO

p\_new\_id;

END

P\_CLIENTE3\_INSERT;

/

</details>

## Funções de Processamento de Dados

Funções foram desenvolvidas para encapsular cálculos e lógicas de negócio.

* F\_TOTAL\_CLIENTES\_POR\_UF(p\_uf VARCHAR2) RETURN NUMBER

Conta o número de clientes únicos que possuem endereço em um determinado estado (UF).

* F\_CALCULAR\_IDADE\_CLIENTE(p\_id\_cliente NUMBER) RETURN NUMBER

Calcula a idade de um cliente com base em sua data de nascimento (armazenada como VARCHAR2).

<details> <summary>💻 Exemplo de Função</summary>

--

Calcula a idade de um cliente.

CREATE

OR

REPLACE

FUNCTION

F\_CALCULAR\_IDADE\_CLIENTE

(

p\_id\_cliente

IN

NUMBER

)

RETURN

NUMBER

IS

v\_data\_nasc\_varchar

VARCHAR2(10);

v\_data\_nasc\_date

DATE;

v\_idade

NUMBER;

BEGIN

SELECT

DATA\_NASCIMENTO

INTO

v\_data\_nasc\_varchar

FROM

TB\_CLIENTE3

WHERE

ID\_CLIENTE

=

p\_id\_cliente;

v\_data\_nasc\_date

:=

TO\_DATE(v\_data\_nasc\_varchar,

'DD/MM/YYYY'

)

;

v\_idade

:=

TRUNC(MONTHS\_BETWEEN(SYSDATE,

v\_data\_nasc\_date)

/

12)

;

RETURN

v\_idade;

EXCEPTION

WHEN

OTHERS

THEN

RETURN

NULL

;

END

F\_CALCULAR\_IDADE\_CLIENTE;

/

</details>

# Consultas Complexas (Relatórios SQL) 📊

Foram desenvolvidas 5 consultas complexas para extrair informações gerenciais, utilizando JOIN, GROUP BY, HAVING, subqueries e funções de agregação:

* 🗺 **Relatório de Clientes por Estado (UF):** Ranking de estados por número de clientes.
* **Relatório de Contatos por Cliente:** Quantidade de contatos registrados para cada cliente.
* **Relatório de Eventos de Alto Impacto:** Identifica eventos EONET que afetaram múltiplos endereços.
* **Relatório de Perfil Detalhado do Cliente:** Usa subqueries para exibir um perfil consolidado.
* 🏙 **Análise de Média de Clientes por Cidade em SP:** Calcula a média de clientes por cidade em um estado específico.

<details> <summary>📈 Exemplo de Relatório (Eventos de Alto Impacto)</summary>

--

Identifica eventos que afetaram um ou mais endereços cadastrados.

SELECT

evt.EONET\_ID,

--

Aplicamos MAX() ao título. Como só há um título por evento, o

resultado é o mesmo.

MAX

(

JSON\_VALUE(evt.JSON,

'$.title'

))

AS

titulo\_do\_evento,

COUNT

(

ee.TB\_ENDERECO3\_ID\_ENDERECO

)

AS

enderecos\_afetados

FROM

TB\_EONET3

evt

JOIN

TB\_ENDERECOEVENTOS3

ee

ON

evt.ID\_EONET

=

ee.TB\_EONET3\_ID\_EONET

GROUP

BY

--

Agrupamos pelo identificador único do evento para corrigir o erro

ORA-00979.

evt.ID\_EONET,

evt.EONET\_ID

HAVING

COUNT

(

ee.TB\_ENDERECO3\_ID\_ENDERECO

)

>

0

ORDER

BY

enderecos\_afetados

DESC

;

</details>

# Blocos Anônimos e Cursores 💻

Blocos PL/SQL foram criados para demonstrar lógicas de negócio e o uso de cursores.

## Lógica de Negócio com Blocos Anônimos

* **Análise Regional:** Um bloco que gera um relatório de clientes por estado, usando LOOP e IF/ ELSE para destacar regiões com alta concentração.
* **Análise de Risco:** Um bloco que percorre clientes e verifica a associação com eventos naturais, emitindo alertas.

## Demonstração de Cursor Explícito

Foi criado um bloco anônimo para demonstrar o controle manual de um cursor com OPEN, FETCH e CLOSE para processar dados linha a linha, gerando um relatório de clientes.

<details> <summary>🖱 Exemplo de Cursor Explícito</summary>

DECLARE

--

Declaração das variáveis que receberão os dados

v\_nome\_completo

VARCHAR2(201);

v\_documento

TB\_CLIENTE3.DOCUMENTO%

TYPE

;

--

Declaração do CURSOR EXPLÍCITO

CURSOR

c\_lista\_clientes

IS

SELECT

NOME

||

' '

||

SOBRENOME,

DOCUMENTO

FROM

TB\_CLIENTE3

ORDER

BY

NOME;

BEGIN

--

1. OPEN: Abre o cursor

OPEN

c\_lista\_clientes;

--

2. LOOP e FETCH: Busca os dados

LOOP

FETCH

c\_lista\_clientes

INTO

v\_nome\_completo,

v\_documento;

EXIT

WHEN

c\_lista\_clientes%NOTFOUND;

--

Processa os dados

DBMS\_OUTPUT.PUT\_LINE(RPAD(v\_nome\_completo,

40)

||

v\_documento);

END

LOOP;

--

3. CLOSE: Fecha o cursor

CLOSE

c\_lista\_clientes;

END

;

/

</details>

1. **Link do Repositório:**

[GitHub](https://github.com/carmipa/GS_FIAP_2025_1SM/tree/main/Mastering_Relational_and_Non_Relational_Database)

1. **Tecnologias Utilizadas:**

Oracle Database

PL/SQL  
  
-------------------

**Imagens da execução do arquivo sql:  
  
Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.Texto

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.**

1. **. Visão Geral 🌍**

   O banco de dados foi desenvolvido para centralizar e gerenciar informações cruciais sobre clientes, incluindo seus dados pessoais, múltiplos contatos e endereços. O sistema integra dados de eventos naturais provenientes do **EONET (Earth Observatory Natural Event Tracker)**, permitindo associar esses eventos a localizações geográficas e analisar o impacto em relação aos clientes. A arquitetura é normalizada, garantindo consistência e minimizando a redundância de dados. [↑](#footnote-ref-1)