GEF ↪ Mastering Relational and Non-Relational Database

*

→ Integrantes

- Eduardo Henrique Strapazzon Nagado - RM558158
- Felipe Silva Maciel - RM555307
- Gustavo Ramires Lazzuri - RM556772

→ Sumário

- Código-Fonte da Tabela de Auditoria e do Trigger
- Testes de Execução

Código da 2ª Sprint Corrigido

→ Introdução

Este documento detalha a implementação do banco de dados para o projeto **Mottu GEF**, um sistema de gestão de frotas de motocicletas inspirado em um cenário operacional real da Mottu.

O foco desta entrega é demonstrar a construção de um ecossistema de dados que vai além da simples criação de tabelas. A arquitetura relacional foi projetada para garantir a integridade e a consistência dos dados de entidades críticas como Pátios, Motos, Operadores e Manutenções.

O diferencial deste projeto reside na implementação de lógica de negócio e automação diretamente no banco de dados através de objetos PL/SQL avançados:

- **Procedures:** Foram desenvolvidas para gerar relatórios complexos, como o PRC_RELATORIO_CUSTOS_MANUAL que agrega dados de múltiplas tabelas, e o PRC_LISTAR_DETECCOES_JSON, que prepara os dados para serem consumidos por APIs modernas.
- **Functions:** Encapsulam regras de negócio essenciais, como a FNC_VERIFICAR_VAGA_PATIO, e fornecem utilitários poderosos como a FNC_RELACIONAL_PARA_JSON, que converte dinamicamente qualquer consulta em formato JSON.
- **Triggers:** Automatizam tarefas cruciais de segurança e rastreabilidade, como o TRG_AUDITA_MOTO, que cria um registro de auditoria imutável para todas as alterações realizadas na tabela de motocicletas.

Em suma, este script demonstra a construção de uma base de dados que não apenas armazena informações, mas também executa regras de negócio, garante a segurança dos dados e oferece uma plataforma de dados pronta para integração com as demais camadas de um sistema de software completo.

→ Função 1: Conversor Relacional para JSON

Objetivo: Criar uma função que recebe um cursor com dados relacionais e os converte para uma string em formato JSON, sem utilizar nenhuma função *built-in* do Oracle para JSON.

Código-Fonte:

```sql
-- FUNÇÃO 1: Converte um SYS_REFCURSOR para JSON CLOB
CREATE OR REPLACE FUNCTION FNC_RELACIONAL_PARA_JSON (
    p_cursor IN SYS_REFCURSOR
) RETURN CLOB
IS
    v_json_clob CLOB;
    v_col_count INTEGER;
    v_desc_tab DBMS_SQL.DESC_TAB;
    v_cursor_id INTEGER;
    v_col_value VARCHAR2(4000);
    v_is_first_row BOOLEAN := TRUE;
    l_cursor SYS_REFCURSOR;
BEGIN
    l_cursor := p_cursor;
    v_cursor_id := DBMS_SQL.TO_CURSOR_NUMBER(l_cursor);
    DBMS_SQL.DESCRIBE_COLUMNS(v_cursor_id, v_col_count, v_desc_tab);

    FOR i IN 1..v_col_count LOOP
        DBMS_SQL.DEFINE_COLUMN(v_cursor_id, i, v_col_value, 4000);
    END LOOP;

    v_json_clob := '[';
    WHILE DBMS_SQL.FETCH_ROWS(v_cursor_id) > 0 LOOP
        IF NOT v_is_first_row THEN v_json_clob := v_json_clob || ','; END IF;
        v_json_clob := v_json_clob || CHR(10) || '{';
        FOR i IN 1..v_col_count LOOP
            DBMS_SQL.COLUMN_VALUE(v_cursor_id, i, v_col_value);
            v_json_clob := v_json_clob || '"' || LOWER(v_desc_tab(i).col_name) || '":"' || REPLACE(v_col_value, '"', '\"') || '"';
            IF i < v_col_count THEN v_json_clob := v_json_clob || ','; END IF;
        END LOOP;
        v_json_clob := v_json_clob || '}';
        v_is_first_row := FALSE;
    END LOOP;
    v_json_clob := v_json_clob || CHR(10) || ']';

    DBMS_SQL.CLOSE_CURSOR(v_cursor_id);
    IF v_is_first_row THEN RETURN '[]'; END IF;
    RETURN v_json_clob;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        IF DBMS_SQL.IS_OPEN(v_cursor_id) THEN DBMS_SQL.CLOSE_CURSOR(v_cursor_id); END IF;
        RETURN '{"erro": "Nenhum dado encontrado no cursor."}';
    WHEN INVALID_CURSOR THEN
        IF DBMS_SQL.IS_OPEN(v_cursor_id) THEN DBMS_SQL.CLOSE_CURSOR(v_cursor_id); END IF;
        RETURN '{"erro": "Cursor invalido ou fechado fornecido."}';
    WHEN OTHERS THEN
        IF DBMS_SQL.IS_OPEN(v_cursor_id) THEN DBMS_SQL.CLOSE_CURSOR(v_cursor_id); END IF;
        RETURN '{"erro": "Ocorreu um erro ao converter para JSON: ' || SQLERRM || '"}';
END FNC_RELACIONAL_PARA_JSON;
/
```

Testes de Execução:

```
--- Relatorio de Deteccoes em formato JSON ---
[
{"deteccao_id":"5","placa":"MNO7890","modelo":"Mottu Sport","tipo_evento":"SAIDA","timestamp_deteccao":"15/10/25 21:26:57,465865","confianca_yolo":",91"},
{"deteccao_id":"4","placa":"JKL3456","modelo":"Mottu Pop","tipo_evento":"ENTRADA","timestamp_deteccao":"15/10/25 21:26:57,464643","confianca_yolo":",9"},
{"deteccao_id":"3","placa":"GHI9012","modelo":"Mottu-E","tipo_evento":"ENTRADA","timestamp_deteccao":"15/10/25 21:26:57,463431","confianca_yolo":",88"},
{"deteccao_id":"2","placa":"DEF5678","modelo":"Mottu Sport","tipo_evento":"SAIDA","timestamp_deteccao":"15/10/25 21:26:57,462133","confianca_yolo":",92"},
{"deteccao_id":"1","placa":"ABC1234","modelo":"Mottu Pop","tipo_evento":"ENTRADA","timestamp_deteccao":"15/10/25 21:26:57,460015","confianca_yolo":",95"}
]
```

2. Procedimento 1: Listagem Detecções com JOIN em JSON (PRC_LISTAR_DETECCOES_JSON)

Objetivo: Este procedimento consulta o histórico de detecções de motocicletas no banco de dados e converte o resultado para o formato JSON. Em seguida, ele exibe esses dados como um relatório no console, tratando possíveis erros, como a ausência de registros.

Código-Fonte:

```
CREATE OR REPLACE PROCEDURE PRC_LISTAR_DETECCOES_JSON
IS
    v_cursor SYS_REFCURSOR;
    v_json_result CLOB;
    e_nenhuma_deteccao EXCEPTION;
BEGIN
    OPEN v_cursor FOR
        SELECT d.DETECCAO_ID, m.PLACA, m.MODELO, d.TIPO_EVENTO, d.TIMESTAMP_DETECCAO, d.CONFIANCA_YOLO
        FROM DETECCAO d
        JOIN MOTO m ON d.MOTO_ID = m.MOTO_ID
        ORDER BY d.TIMESTAMP_DETECCAO DESC;

    v_json_result := FNC_RELACIONAL_PARA_JSON(v_cursor);

    IF v_json_result = '[]' THEN
        RAISE e_nenhuma_deteccao;
    END IF;

    DBMS_OUTPUT.PUT_LINE('--- Relatorio de Deteccoes em formato JSON ---');
    DBMS_OUTPUT.PUT_LINE(v_json_result);
    DBMS_OUTPUT.PUT_LINE('-------------------------------------------');
EXCEPTION
    WHEN e_nenhuma_deteccao THEN DBMS_OUTPUT.PUT_LINE('Erro Tratado: Nenhuma deteccao encontrada para gerar o relatorio.');
    WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Erro Tratado: A consulta nao retornou nenhum dado (NO_DATA_FOUND).');
    WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Erro Tratado Inesperado no procedimento PRC_LISTAR_DETECCOES_JSON: ' || SQLERRM);
END PRC_LISTAR_DETECCOES_JSON;
/
```

Testes de Execução:

```
--- Relatorio de Deteccoes em formato JSON ---
[
{"deteccao_id":"5","placa":"MNO7890","modelo":"Mottu Sport","tipo_evento":"SAIDA","timestamp_deteccao":"15/10/25 21:26:57,465865","confianca_yolo":",91"},
{"deteccao_id":"4","placa":"JKL3456","modelo":"Mottu Pop","tipo_evento":"ENTRADA","timestamp_deteccao":"15/10/25 21:26:57,464643","confianca_yolo":",9"},
{"deteccao_id":"3","placa":"GHI9012","modelo":"Mottu-E","tipo_evento":"ENTRADA","timestamp_deteccao":"15/10/25 21:26:57,463431","confianca_yolo":",88"},
{"deteccao_id":"2","placa":"DEF5678","modelo":"Mottu Sport","tipo_evento":"SAIDA","timestamp_deteccao":"15/10/25 21:26:57,462133","confianca_yolo":",92"},
{"deteccao_id":"1","placa":"ABC1234","modelo":"Mottu Pop","tipo_evento":"ENTRADA","timestamp_deteccao":"15/10/25 21:26:57,460015","confianca_yolo":",95"}
]
```

```
TESTE 7: Exceção em PRC_LISTAR_DETECCOES_JSON - Cursor invalido
Resultado: {"erro": "Cursor invalido ou fechado fornecido."}
```

3. Função 2: Verificação de Vagas no Pátio (FNC_VERIFICAR_VAGA_PATIO)

Objetivo: Esta função verifica se um pátio específico possui motos "Prontas para aluguel". Ela retorna 1 se encontrar alguma e 0 se não encontrar, além de tratar erros para pátios inválidos ou não existentes.

Código-Fonte:

```
-- FUNÇÃO 2: Verifica disponibilidade de vagas no pátio e motos prontas para aluguel
CREATE OR REPLACE FUNCTION FNC_VERIFICAR_VAGA_PATIO (p_patio_id IN NUMBER) RETURN NUMBER
IS
    v_capacidade NUMBER;
    v_motos_prontas NUMBER;
    e_capacidade_invalida EXCEPTION;
BEGIN
    SELECT CAPACIDADE INTO v_capacidade FROM PATIO WHERE PATIO_ID = p_patio_id;
    IF v_capacidade IS NULL OR v_capacidade <= 0 THEN
        RAISE e_capacidade_invalida;
    END IF;

    SELECT COUNT(*) INTO v_motos_prontas FROM MOTO WHERE PATIO_ID = p_patio_id AND STATUS = 'Pronta para aluguel';

    IF v_motos_prontas > 0 THEN
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20002, 'Erro Tratado: Patio com ID ' || p_patio_id || ' nao encontrado.');
    WHEN e_capacidade_invalida THEN RAISE_APPLICATION_ERROR(-20003, 'Erro Tratado: A capacidade do patio ' || p_patio_id || ' nao e valida.');
    WHEN OTHERS THEN RAISE_APPLICATION_ERROR(-20004, 'Erro Tratado Inesperado ao verificar vagas: ' || SQLERRM);
END FNC_VERIFICAR_VAGA_PATIO;
/
```

Testes de Execução:

```
TESTE 4: Funcao FNC_VERIFICAR_VAGA_PATIO - Patio 1
Resultado: Ha motos prontas para aluguel no Patio 1.
```

```
TESTE 6: Exceção em FNC_VERIFICAR_VAGA_PATIO - Patio inexistente (ID 999)
Exceção capturada corretamente: ORA-20002: Erro Tratado: Patio com ID 999 nao encontrado.
```

4. Procedimento 2: Relatório de Custos de Manutenção (PRC_RELATORIO_CUSTOS_MANUAL)

Objetivo: Este procedimento gera um relatório de custos de manutenção, agrupando os dados por pátio. Ele exibe os custos, calcula e imprime os subtotais para cada pátio e, ao final, apresenta o custo total geral.

Código-Fonte:

```
CREATE OR REPLACE PROCEDURE PRC_RELATORIO_CUSTOS_MANUAL
IS
    CURSOR c_manutencoes IS
        SELECT p.NOME AS nome_patio, o.CARGO, m.CUSTO
        FROM MANUTENCAO m
        JOIN OPERADOR o ON m.OPERADOR_ID = o.OPERADOR_ID
        JOIN PATIO p ON o.PATIO_ID = p.PATIO_ID
        ORDER BY p.NOME, o.CARGO;

    v_patio_atual PATIO.NOME%TYPE := NULL;
    v_subtotal_patio NUMBER(12, 2) := 0;
    v_total_geral NUMBER(14, 2) := 0;
    v_primeira_linha BOOLEAN := TRUE;
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Relatorio de Custos de Manutencao por Patio e Cargo ---');
    DBMS_OUTPUT.PUT_LINE(RPAD('Patio', 30) || RPAD('Cargo', 20) || 'Custo (R$)');
    DBMS_OUTPUT.PUT_LINE(RPAD('-', 70, '-'));

    FOR rec IN c_manutencoes LOOP
        IF v_patio_atual IS NOT NULL AND rec.nome_patio != v_patio_atual THEN
            DBMS_OUTPUT.PUT_LINE(RPAD('-', 70, '-'));
            DBMS_OUTPUT.PUT_LINE(RPAD('Subtotal ' || v_patio_atual, 50) || TO_CHAR(v_subtotal_patio, '999G999D99'));
            DBMS_OUTPUT.PUT_LINE('');
            v_total_geral := v_total_geral + v_subtotal_patio;
            v_subtotal_patio := 0;
        END IF;

        DBMS_OUTPUT.PUT_LINE(RPAD(rec.nome_patio, 30) || RPAD(rec.cargo, 20) || TO_CHAR(rec.CUSTO, '999G999D99'));
        v_subtotal_patio := v_subtotal_patio + rec.CUSTO;
        v_patio_atual := rec.nome_patio;
        v_primeira_linha := FALSE;
    END LOOP;

    IF NOT v_primeira_linha THEN
        DBMS_OUTPUT.PUT_LINE(RPAD('-', 70, '-'));
        DBMS_OUTPUT.PUT_LINE(RPAD('Subtotal ' || v_patio_atual, 50) || TO_CHAR(v_subtotal_patio, '999G999D99'));
        v_total_geral := v_total_geral + v_subtotal_patio;
        DBMS_OUTPUT.PUT_LINE(RPAD('=', 70, '='));
        DBMS_OUTPUT.PUT_LINE(RPAD('TOTAL GERAL', 50) || TO_CHAR(v_total_geral, '999G999D99'));
    ELSE
        DBMS_OUTPUT.PUT_LINE('Nenhuma manutencao encontrada para o relatorio.');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Erro Tratado: Nenhuma manutencao foi encontrada para o relatorio.');
    WHEN VALUE_ERROR THEN
        DBMS_OUTPUT.PUT_LINE('Erro Tratado: Ocorreu um erro de conversao de dados no relatorio de custos.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erro Tratado Inesperado ao gerar relatorio de custos: ' || SQLERRM);
END PRC_RELATORIO_CUSTOS_MANUAL;
/
```

Testes de Execução:

```
TESTE 5: Procedimento PRC_RELATORIO_CUSTOS_MANUAL
--- Relatorio de Custos de Manutencao por Patio e Cargo ---
Patio                          Cargo                Custo (R$)
------------------------------------------------------------------------
Pátio Zona Leste               TECNICO                   150,00
Pátio Zona Leste               TECNICO                   200,00
Pátio Zona Leste               TECNICO                   300,00
Pátio Zona Leste               TECNICO                   120,00
Pátio Zona Leste               TECNICO                   100,00
------------------------------------------------------------------------
Subtotal Pátio Zona Leste                                870,00
========================================================================
TOTAL GERAL                                              870,00
```

```
TESTE 8: Exceção em PRC_RELATORIO_CUSTOS_MANUAL - Nenhuma manutencao
--- Relatorio de Custos de Manutencao por Patio e Cargo ---
Patio                          Cargo                Custo (R$)
------------------------------------------------------------------------
Nenhuma manutencao encontrada para o relatorio.
Rollback executado - dados restaurados.
```

5. Trigger de Auditoria em Motocicletas (TRG_AUDITA_MOTO)

Objetivo: Criar um trigger que audita todas as operações de INSERT, UPDATE e DELETE na tabela MOTO.

Código-Fonte:

```sql
-- TRIGGER: Auditoria de operações DML na tabela MOTO
CREATE OR REPLACE TRIGGER TRG_AUDITA_MOTO
AFTER INSERT OR UPDATE OR DELETE ON MOTO
FOR EACH ROW
DECLARE
    v_old_values CLOB;
    v_new_values CLOB;
BEGIN
    IF DELETING OR UPDATING THEN
        v_old_values := 'MOTO_ID=' || :OLD.MOTO_ID || ', PLACA=' || :OLD.PLACA || ', MODELO=' || :OLD.MODELO || ', ANO=' || :OLD.ANO || ', STATUS=' || :OLD.STATUS;
    END IF;
    IF INSERTING OR UPDATING THEN
        v_new_values := 'MOTO_ID=' || :NEW.MOTO_ID || ', PLACA=' || :NEW.PLACA || ', MODELO=' || :NEW.MODELO || ', ANO=' || :NEW.ANO || ', STATUS=' || :NEW.STATUS;
    END IF;

    IF INSERTING THEN
        INSERT INTO AUDITORIA (NOME_USUARIO, TIPO_OPERACAO, DATA_HORA, VALORES_NOVOS) VALUES (USER, 'INSERT', SYSTIMESTAMP, v_new_values);
    ELSIF UPDATING THEN
        INSERT INTO AUDITORIA (NOME_USUARIO, TIPO_OPERACAO, DATA_HORA, VALORES_ANTERIORES, VALORES_NOVOS) VALUES (USER, 'UPDATE', SYSTIMESTAMP, v_old_values, v_new_values);
    ELSIF DELETING THEN
        INSERT INTO AUDITORIA (NOME_USUARIO, TIPO_OPERACAO, DATA_HORA, VALORES_ANTERIORES) VALUES (USER, 'DELETE', SYSTIMESTAMP, v_old_values);
    END IF;
EXCEPTION
    WHEN OTHERS THEN RAISE_APPLICATION_ERROR(-20001, 'Erro no trigger de auditoria: ' || SQLERRM);
END TRG_AUDITA_MOTO;
/
```

Testes de Execução:

```
TESTE 2: Trigger TRG_AUDITA_MOTO - Testando INSERT
Moto de teste inserida. Auditoria registrada.
```

```
TESTE 3: Trigger TRG_AUDITA_MOTO - Testando UPDATE
Moto de teste atualizada. Auditoria registrada.
```

```
TESTE 9: Trigger TRG_AUDITA_MOTO - Testando DELETE
Moto de teste deletada. Auditoria registrada.

TESTE 10: Verificando registros na tabela AUDITORIA
Operacao: DELETE | Hora: 15/09/25 21:26:58,509142
Operacao: UPDATE | Hora: 15/09/25 21:26:58,474172
Operacao: INSERT | Hora: 15/09/25 21:26:58,468458
```

6. 2ª Sprint Corrigida

Na Sprint 2, a entrega foi parcial por não incluir a documentação completa do modelo de dados e um script SQL unificado. Para a Sprint 3, corrigimos essas falhas: o modelo lógico foi revisado e documentado, e todas as tabelas, como MOTO, PATIO e DETECCAO, foram recriadas e populadas. O script SQL foi consolidado e agora inclui funcionalidades robustas como a trigger de auditoria TRG_AUDITA_MOTO, procedures para geração de relatórios (PRC_LISTAR_DETECCOES_JSON), e funções de validação como a FNC_VERIFICAR_VAGA_PATIO, todas com tratamento de exceções. A documentação foi enriquecida com prints de teste que comprovam a execução e o funcionamento de cada objeto.

Criação das Tabelas e Sequências

```
-- Tabela PATIO: Representa os pátios de armazenamento de motocicletas
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE PATIO (
        PATIO_ID NUMBER PRIMARY KEY,
        NOME VARCHAR2(255) NOT NULL,
        LOCALIZACAO VARCHAR2(255) NOT NULL,
        CAPACIDADE NUMBER(5) DEFAULT 100 NOT NULL
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela PATIO criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela PATIO já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE patio_seq START WITH 1 INCREMENT BY 1 NOCACHE;

-- Tabela MOTO: Registra todas as motocicletas do sistema
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE MOTO (
        MOTO_ID NUMBER PRIMARY KEY,
        PLACA VARCHAR2(10) UNIQUE NOT NULL,
        MODELO VARCHAR2(50) NOT NULL CHECK (MODELO IN (''Mottu Pop'', ''Mottu Sport'', ''Mottu-E'')),
        ANO NUMBER(4) CHECK (ANO >= 2015 AND ANO <= 2050),
        PATIO_ID NUMBER NOT NULL,
        STATUS VARCHAR2(30) CHECK (STATUS IN (''Pronta para aluguel'', ''Em manutencao'', ''Em quarentena'', ''Alta prioridade'', ''Reservada'', ''Aguardando vistoria'')),
        DATA_CADASTRO TIMESTAMP DEFAULT SYSTIMESTAMP,
        CONSTRAINT FK_MOTO_PATIO FOREIGN KEY (PATIO_ID) REFERENCES PATIO(PATIO_ID)
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela MOTO criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela MOTO já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE moto_seq START WITH 1 INCREMENT BY 1 NOCACHE;

-- Tabela DETECCAO: Registra eventos de detecção via IoT (YOLO + OCR)
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE DETECCAO (
        DETECCAO_ID NUMBER PRIMARY KEY,
        MOTO_ID NUMBER NOT NULL,
        PATIO_ID NUMBER NOT NULL,
        TIPO_EVENTO VARCHAR2(20) CHECK (TIPO_EVENTO IN (''ENTRADA'', ''SAIDA'')),
        TIMESTAMP_DETECCAO TIMESTAMP DEFAULT SYSTIMESTAMP,
        CONFIANCA_YOLO NUMBER(3,2) CHECK (CONFIANCA_YOLO >= 0 AND CONFIANCA_YOLO <= 1),
        CONSTRAINT FK_DETECCAO_MOTO FOREIGN KEY (MOTO_ID) REFERENCES MOTO(MOTO_ID),
        CONSTRAINT FK_DETECCAO_PATIO FOREIGN KEY (PATIO_ID) REFERENCES PATIO(PATIO_ID)
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela DETECCAO criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela DETECCAO já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE deteccao_seq START WITH 1 INCREMENT BY 1 NOCACHE;
```

```
-- Tabela OPERADOR: Funcionários que gerenciam os pátios
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE OPERADOR (
        OPERADOR_ID NUMBER PRIMARY KEY,
        NOME VARCHAR2(255) NOT NULL,
        EMAIL VARCHAR2(255) NOT NULL UNIQUE,
        PASSWORD VARCHAR2(255) NOT NULL,
        CARGO VARCHAR2(50) NOT NULL CHECK (CARGO IN (''GESTOR'', ''OPERADOR'', ''TECNICO'')),
        PATIO_ID NUMBER NOT NULL,
        SALARIO NUMBER(10, 2) DEFAULT 2500.00 NOT NULL,
        DATA_ADMISSAO DATE DEFAULT SYSDATE,
        CONSTRAINT FK_OPERADOR_PATIO FOREIGN KEY (PATIO_ID) REFERENCES PATIO(PATIO_ID)
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela OPERADOR criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela OPERADOR já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE operador_seq START WITH 1 INCREMENT BY 1 NOCACHE;

-- Tabela MANUTENCAO: Registra eventos de manutenção das motocicletas
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE MANUTENCAO (
        MANUTENCAO_ID NUMBER PRIMARY KEY,
        MOTO_ID NUMBER NOT NULL,
        OPERADOR_ID NUMBER NOT NULL,
        TIPO_SERVICO VARCHAR2(100) NOT NULL,
        DATA_INICIO TIMESTAMP DEFAULT SYSTIMESTAMP,
        DATA_FIM TIMESTAMP,
        CUSTO NUMBER(10, 2) DEFAULT 0,
        CONSTRAINT FK_MANUTENCAO_MOTO FOREIGN KEY (MOTO_ID) REFERENCES MOTO(MOTO_ID),
        CONSTRAINT FK_MANUTENCAO_OPERADOR FOREIGN KEY (OPERADOR_ID) REFERENCES OPERADOR(OPERADOR_ID)
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela MANUTENCAO criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela MANUTENCAO já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE manutencao_seq START WITH 1 INCREMENT BY 1 NOCACHE;
```

Outputs:

```
========== EXECUTANDO TESTES DA SPRINT 3 ==========

--- TESTES DE SUCESSO ---

TESTE 1: Procedimento PRC_LISTAR_DETECCOES_JSON
--- Relatorio de Deteccoes em formato JSON ---
[
{"deteccao_id":"5","placa":"MNO7890","modelo":"Mottu Sport","tipo_evento":"SAIDA","timestamp_deteccao":"15/10/25 21:26:57,465865","confianca_yolo":",91"},
{"deteccao_id":"4","placa":"JKL3456","modelo":"Mottu Pop","tipo_evento":"ENTRADA","timestamp_deteccao":"15/10/25 21:26:57,464643","confianca_yolo":",9"},
{"deteccao_id":"3","placa":"GHI9012","modelo":"Mottu-E","tipo_evento":"ENTRADA","timestamp_deteccao":"15/10/25 21:26:57,463431","confianca_yolo":",88"},
{"deteccao_id":"2","placa":"DEF5678","modelo":"Mottu Sport","tipo_evento":"SAIDA","timestamp_deteccao":"15/10/25 21:26:57,462133","confianca_yolo":",92"},
{"deteccao_id":"1","placa":"ABC1234","modelo":"Mottu Pop","tipo_evento":"ENTRADA","timestamp_deteccao":"15/10/25 21:26:57,460015","confianca_yolo":",95"}
]
--------------------------------------------

TESTE 2: Trigger TRG_AUDITA_MOTO - Testando INSERT
Moto de teste inserida. Auditoria registrada.

TESTE 3: Trigger TRG_AUDITA_MOTO - Testando UPDATE
Moto de teste atualizada. Auditoria registrada.

TESTE 4: Funcao FNC_VERIFICAR_VAGA_PATIO - Patio 1
Resultado: Ha motos prontas para aluguel no Patio 1.

TESTE 5: Procedimento PRC_RELATORIO_CUSTOS_MANUAL
--- Relatorio de Custos de Manutencao por Patio e Cargo ---
Patio                          Cargo               Custo (R$)
----------------------------------------------------------------------
Pátio Zona Leste               TECNICO                  150,00
Pátio Zona Leste               TECNICO                  200,00
Pátio Zona Leste               TECNICO                  300,00
Pátio Zona Leste               TECNICO                  120,00
Pátio Zona Leste               TECNICO                  100,00
----------------------------------------------------------------------
Subtotal Pátio Zona Leste                               870,00
======================================================================
TOTAL GERAL                                             870,00

--- TESTES DE EXCEÇÃO ---

TESTE 6: Exceção em FNC_VERIFICAR_VAGA_PATIO - Patio inexistente (ID 999)
Exceção capturada corretamente: ORA-20002: Erro Tratado: Patio com ID 999 nao encontrado.

TESTE 7: Exceção em PRC_LISTAR_DETECCOES_JSON - Cursor invalido
Resultado: {"erro": "Cursor invalido ou fechado fornecido."}

TESTE 8: Exceção em PRC_RELATORIO_CUSTOS_MANUAL - Nenhuma manutencao
--- Relatorio de Custos de Manutencao por Patio e Cargo ---
Patio                          Cargo               Custo (R$)
----------------------------------------------------------------------
Nenhuma manutencao encontrada para o relatorio.
Rollback executado - dados restaurados.

TESTE 9: Trigger TRG_AUDITA_MOTO - Testando DELETE
Moto de teste deletada. Auditoria registrada.

TESTE 10: Verificando registros na tabela AUDITORIA
Operacao: DELETE | Hora: 15/10/25 21:26:58,509142
Operacao: UPDATE | Hora: 15/10/25 21:26:58,474172
Operacao: INSERT | Hora: 15/10/25 21:26:58,468458

========== TESTES FINALIZADOS COM SUCESSO ==========
```