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

→ 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:

```
-- FUNÇÃO 1: Converte um SYS_REFCURSOR para uma string no formato JSON.
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;
    -- Variável local para receber a referência do cursor, permitindo sua manipulação pelo pacote DBMS_SQL.
    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."}';
    -- Trata a exceção que ocorre se o cursor de entrada for inválido ou estiver fechado.
    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 inválido 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:

```
--- Relatório de Pacientes em formato JSON ---
[
{"nome_paciente":"Ana Silva","idade":"75","nome_abrigo":"Abrigo Esperança","endereco_abrigo":"Rua das Flores, 100"},
{"nome_paciente":"Maria Oliveira","idade":"82","nome_abrigo":"Abrigo Esperança","endereco_abrigo":"Rua das Flores, 100"},
{"nome_paciente":"Carlos Souza","idade":"68","nome_abrigo":"Abrigo Solidário","endereco_abrigo":"Avenida Principal, 250"},
{"nome_paciente":"Fernanda Lima","idade":"60","nome_abrigo":"Abrigo Solidário","endereco_abrigo":"Avenida Principal, 250"},
{"nome_paciente":"João Santos","idade":"91","nome_abrigo":"Abrigo do Povo","endereco_abrigo":"Praça da Liberdade, 15"}
]
-----------------------------------------
```

```
Teste de Exceção (Função 1): Passando um cursor inválido/fechado.
{"erro": "Cursor inválido ou fechado fornecido."}
```

2. Procedimento 1: Listagem de Pacientes com JOIN em JSON (PRC_LISTAR_PACIENTES_JSON)

Objetivo: Realizar uma consulta com JOIN entre as tabelas PACIENTES e ABRIGO e exibir o resultado em formato JSON, utilizando a FNC_RELACIONAL_PARA_JSON.

Código-Fonte:

```sql
-- PROCEDIMENTO 1: Lista pacientes e seus abrigos, retornando o resultado em formato JSON.
CREATE OR REPLACE PROCEDURE PRC_LISTAR_PACIENTES_JSON
IS
    v_cursor SYS_REFCURSOR;
    v_json_result CLOB;
    e_nenhum_paciente EXCEPTION;
BEGIN
    OPEN v_cursor FOR
        SELECT p.NOME AS nome_paciente, p.IDADE, a.NOME AS nome_abrigo, a.ENDERECO AS endereco_abrigo
        FROM PACIENTES p JOIN ABRIGO a ON p.ABRIGO_ID = a.ABRIGOID;

    v_json_result := FNC_RELACIONAL_PARA_JSON(v_cursor);

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

    DBMS_OUTPUT.PUT_LINE('--- Relatório de Pacientes em formato JSON ---');
    DBMS_OUTPUT.PUT_LINE(v_json_result);
    DBMS_OUTPUT.PUT_LINE('-------------------------------------------');
EXCEPTION
    WHEN e_nenhum_paciente THEN DBMS_OUTPUT.PUT_LINE('Erro Tratado: Nenhum paciente encontrado para gerar o relatório.');
    WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Erro Tratado: A consulta não retornou nenhum dado (NO_DATA_FOUND).');
    WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE('Erro Tratado Inesperado no procedimento PRC_LISTAR_PACIENTES_JSON: ' || SQLERRM);
END PRC_LISTAR_PACIENTES_JSON;
/
```

Testes de Execução:

```
--- Relatório de Pacientes em formato JSON ---
[
{"nome_paciente":"Ana Silva","idade":"75","nome_abrigo":"Abrigo Esperança","endereco_abrigo":"Rua das Flores, 100"},
{"nome_paciente":"Maria Oliveira","idade":"82","nome_abrigo":"Abrigo Esperança","endereco_abrigo":"Rua das Flores, 100"},
{"nome_paciente":"Carlos Souza","idade":"68","nome_abrigo":"Abrigo Solidário","endereco_abrigo":"Avenida Principal, 250"},
{"nome_paciente":"Fernanda Lima","idade":"60","nome_abrigo":"Abrigo Solidário","endereco_abrigo":"Avenida Principal, 250"},
{"nome_paciente":"João Santos","idade":"91","nome_abrigo":"Abrigo do Povo","endereco_abrigo":"Praça da Liberdade, 15"}
]
-------------------------------------------
```

```
Teste de Exceção (Procedimento 1): Nenhum paciente encontrado.
Erro Tratado: Nenhum paciente encontrado para gerar o relatório.
```

3. Função 2: Verificação de Vagas em Abrigo (FNC_VERIFICAR_VAGA_ABRIGO)

Objetivo: Criar uma função que implementa a lógica de negócio de verificar se um abrigo possui vagas disponíveis.

Código-Fonte:

```
-- FUNÇÃO 2: Verifica a disponibilidade de vagas em um abrigo específico.
CREATE OR REPLACE FUNCTION FNC_VERIFICAR_VAGA_ABRIGO (p_abrigo_id IN NUMBER) RETURN NUMBER
IS
    v_capacidade NUMBER;
    v_pacientes_atuais NUMBER;
    e_capacidade_invalida EXCEPTION;
BEGIN
    SELECT CAPACIDADE INTO v_capacidade FROM ABRIGO WHERE ABRIGOID = p_abrigo_id;
    IF v_capacidade IS NULL OR v_capacidade <= 0 THEN
        RAISE e_capacidade_invalida;
    END IF;

    SELECT COUNT(*) INTO v_pacientes_atuais FROM PACIENTES WHERE ABRIGO_ID = p_abrigo_id;
    -- Retorna 1 se há vaga, 0 se não há vaga.
    IF v_pacientes_atuais < v_capacidade THEN
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20002, 'Erro Tratado: Abrigo com ID ' || p_abrigo_id || ' não encontrado.');
    WHEN e_capacidade_invalida THEN RAISE_APPLICATION_ERROR(-20003, 'Erro Tratado: A capacidade do abrigo ' || p_abrigo_id || ' não é válida.');
    WHEN OTHERS THEN RAISE_APPLICATION_ERROR(-20004, 'Erro Tratado Inesperado ao verificar vagas: ' || SQLERRM);
END FNC_VERIFICAR_VAGA_ABRIGO;
/
```

Testes de Execução:

```
Resultado (Abrigo 1): Há vagas disponíveis.
Teste de Exceção (Abrigo 999): ORA-20002: Erro Tratado: Abrigo com ID 999 não encontrado.
```

4. Procedimento 2: Relatório Salarial com Agregação Manual (PRC_RELATORIO_SALARIAL_MANUAL)

Objetivo: Gerar um relatório de salários com subtotais por abrigo e um total geral, utilizando lógica de agregação manual sem funções automáticas do Oracle.

Código-Fonte:

```
-- PROCEDIMENTO 2: Gera um relatório salarial com subtotais por abrigo e um total geral, com lógica de agregação manual.
CREATE OR REPLACE PROCEDURE PRC_RELATORIO_SALARIAL_MANUAL
IS
    CURSOR c_funcionarios IS
        SELECT a.NOME AS nome_abrigo, f.CARGO, f.SALARIO
        FROM FUNCIONARIO f
        JOIN ABRIGO a ON f.ABRIGO_ID = a.ABRIGOID
        ORDER BY a.NOME, f.CARGO;

    v_abrigo_atual ABRIGO.NOME%TYPE := NULL;
    v_subtotal_abrigo NUMBER(12, 2) := 0;
    v_total_geral NUMBER(14, 2) := 0;
    v_primeira_linha BOOLEAN := TRUE;
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Relatório de Salários por Abrigo e Cargo ---');
    DBMS_OUTPUT.PUT_LINE(RPAD('Abrigo', 30) || RPAD('Cargo', 20) || 'Salário');
    DBMS_OUTPUT.PUT_LINE(RPAD('-', 70, '-'));

    FOR rec IN c_funcionarios LOOP
        -- Se o abrigo do registro atual for diferente do anterior, exibe o subtotal.
        IF v_abrigo_atual IS NOT NULL AND rec.nome_abrigo != v_abrigo_atual THEN
            DBMS_OUTPUT.PUT_LINE(RPAD('-', 70, '-'));
            DBMS_OUTPUT.PUT_LINE(RPAD('Subtotal ' || v_abrigo_atual, 50) || TO_CHAR(v_subtotal_abrigo, '999G999D99'));
            DBMS_OUTPUT.PUT_LINE('');
            v_total_geral := v_total_geral + v_subtotal_abrigo;
            v_subtotal_abrigo := 0;
        END IF;

        DBMS_OUTPUT.PUT_LINE(RPAD(rec.nome_abrigo, 30) || RPAD(rec.cargo, 20) || TO_CHAR(rec.SALARIO, '999G999D99'));
        v_subtotal_abrigo := v_subtotal_abrigo + rec.SALARIO;
        v_abrigo_atual := rec.nome_abrigo;
        v_primeira_linha := FALSE;
    END LOOP;

    -- Exibe o subtotal do último grupo e o total geral após o loop.
    IF NOT v_primeira_linha THEN
        DBMS_OUTPUT.PUT_LINE(RPAD('-', 70, '-'));
        DBMS_OUTPUT.PUT_LINE(RPAD('Subtotal ' || v_abrigo_atual, 50) || TO_CHAR(v_subtotal_abrigo, '999G999D99'));
        v_total_geral := v_total_geral + v_subtotal_abrigo;
        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('Nenhum funcionário encontrado para o relatório.');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Erro Tratado: Nenhum funcionário foi encontrado para o relatório.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erro Tratado Inesperado ao gerar relatório salarial: ' || SQLERRM);
END PRC_RELATORIO_SALARIAL_MANUAL;
/
```

Testes de Execução:

```
--- Relatório de Salários por Abrigo e Cargo ---
Abrigo                          Cargo                Salário
-------------------------------------------------------------------
Abrigo Esperança                ADMINISTRADOR          5,000.00
Abrigo Esperança                FUNCIONARIO            2,500.00
-------------------------------------------------------------------
Subtotal Abrigo Esperança                              7,500.00

Abrigo Solidário                FUNCIONARIO            2,800.00
Abrigo Solidário                VOLUNTARIO             1,500.00
-------------------------------------------------------------------
Subtotal Abrigo Solidário                              4,300.00

Abrigo do Povo                  FUNCIONARIO            1,500.00
-------------------------------------------------------------------
Subtotal Abrigo do Povo                                1,500.00
===================================================================
TOTAL GERAL                                           13,300.00
```

```
Teste de Exceção (Procedimento 2): Nenhum funcionário encontrado.
--- Relatório de Salários por Abrigo e Cargo ---
Abrigo                          Cargo                Salário
-------------------------------------------------------------------
Nenhum funcionário encontrado para o relatório.
```

5. Trigger de Auditoria em PACIENTES (TRG_AUDITA_PACIENTES)

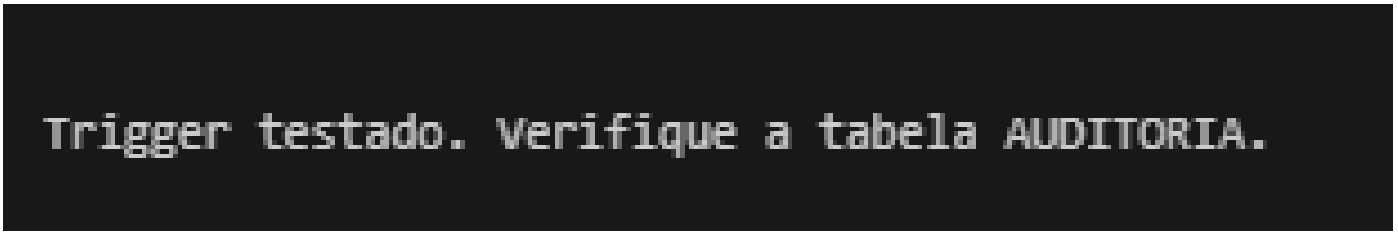
Objetivo: Criar um trigger que audita todas as operações de INSERT, UPDATE e DELETE na tabela PACIENTES.

Código-Fonte:

```sql
-- TRIGGER: Captura operações de DML na tabela PACIENTES e registra na tabela AUDITORIA.
CREATE OR REPLACE TRIGGER TRG_AUDITA_PACIENTES
AFTER INSERT OR UPDATE OR DELETE ON PACIENTES
FOR EACH ROW
DECLARE
    v_old_values CLOB; v_new_values CLOB;
BEGIN
    IF DELETING OR UPDATING THEN
        v_old_values := 'ID=' || :OLD.ID || ', NOME=' || :OLD.NOME || ', IDADE=' || :OLD.IDADE || ', ABRIGO_ID=' || :OLD.ABRIGO_ID;
    END IF;
    IF INSERTING OR UPDATING THEN
        v_new_values := 'ID=' || :NEW.ID || ', NOME=' || :NEW.NOME || ', IDADE=' || :NEW.IDADE || ', ABRIGO_ID=' || :NEW.ABRIGO_ID;
    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_PACIENTES;
/
```

Testes de Execução:



6. 2ª Sprint Corrigida

Na Sprint 2, a entrega foi considerada incompleta por ausência de modelo descritivo, lógico e do SQL com todos os objetos implementados. Já na Sprint 3, esses pontos foram corrigidos: o modelo lógico foi revisado e documentado, todas as tabelas foram recriadas e populadas com dados suficientes, e o script SQL consolidado passou a incluir funções, procedures melhores estruturadas, triggers de auditoria e tratamento de exceções. Além disso, a documentação contém prints de execução e exemplos de uso real, garantindo maior clareza e qualidade no projeto.

Criação das Tabelas e Sequências

```
-- Criação das Tabelas e Sequências

-- Tabela ABRIGO: Armazena informações sobre os abrigos.
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE ABRIGO (
        ABRIGOID NUMBER PRIMARY KEY,
        NOME VARCHAR2(255) NOT NULL,
        ENDERECO VARCHAR2(255) NOT NULL
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela ABRIGO criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela ABRIGO já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE abrigo_seq START WITH 1 INCREMENT BY 1 NOCACHE;

-- Tabela NFC: Armazena os identificadores únicos de NFC.
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE NFC ( ID_NFC NUMBER PRIMARY KEY )';
    DBMS_OUTPUT.PUT_LINE('Tabela NFC criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela NFC já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE nfc_seq START WITH 1 INCREMENT BY 1 NOCACHE;

-- Tabela BATIMENTO_CARDIACO: Armazena dados de batimentos cardíacos coletados por dispositivos IoT.
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE BATIMENTO_CARDIACO (
        IDBATIMENTOCARDIACO VARCHAR2(200) PRIMARY KEY,
        BPM NUMBER(3) CHECK (BPM BETWEEN 30 AND 220),
        TIMESTAMP NUMBER(14) NOT NULL
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela BATIMENTO_CARDIACO criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela BATIMENTO_CARDIACO já existe.'); ELSE RAISE; END IF;
END;
/

-- Tabela PULSEIRA: Tabela de associação que conecta um NFC e um dispositivo IoT a uma pulseira.
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE PULSEIRA (
        ID_PULSEIRA NUMBER PRIMARY KEY,
        NFC_ID NUMBER UNIQUE,
        IOT_ID VARCHAR(200) UNIQUE,
        CONSTRAINT FK_PULSEIRA_NFC FOREIGN KEY (NFC_ID) REFERENCES NFC(ID_NFC),
        CONSTRAINT FK_PULSEIRA_IOT FOREIGN KEY (IOT_ID) REFERENCES BATIMENTO_CARDIACO(IDBATIMENTOCARDIACO)
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela PULSEIRA criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela PULSEIRA já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE pulseira_seq START WITH 1 INCREMENT BY 1 NOCACHE;
```

```
-- Tabela PACIENTES: Armazena os dados dos pacientes, associando-os a um abrigo e uma pulseira.
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE PACIENTES (
        ID NUMBER PRIMARY KEY,
        NOME VARCHAR2(255) NOT NULL,
        IDADE NUMBER CHECK (IDADE BETWEEN 0 AND 130),
        ENDERECO VARCHAR2(255),
        ABRIGO_ID NUMBER NOT NULL,
        PULSEIRA_ID NUMBER UNIQUE NOT NULL,
        CONSTRAINT FK_PACIENTES_ABRIGO FOREIGN KEY (ABRIGO_ID) REFERENCES ABRIGO(ABRIGOID),
        CONSTRAINT FK_PACIENTES_PULSEIRA FOREIGN KEY (PULSEIRA_ID) REFERENCES PULSEIRA(ID_PULSEIRA)
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela PACIENTES criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela PACIENTES já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE paciente_seq START WITH 1 INCREMENT BY 1 NOCACHE;

-- Tabela FUNCIONARIO: Armazena os dados dos funcionários e voluntários, associando-os a um abrigo.
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE FUNCIONARIO (
        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 (''ADMINISTRADOR'', ''FUNCIONARIO'', ''VOLUNTARIO'')),
        ABRIGO_ID NUMBER NOT NULL,
        CONSTRAINT FK_FUNCIONARIO_ABRIGO FOREIGN KEY (ABRIGO_ID) REFERENCES ABRIGO(ABRIGOID)
    )';
    DBMS_OUTPUT.PUT_LINE('Tabela FUNCIONARIO criada.');
EXCEPTION WHEN OTHERS THEN IF SQLCODE = -955 THEN DBMS_OUTPUT.PUT_LINE('Tabela FUNCIONARIO já existe.'); ELSE RAISE; END IF;
END;
/
CREATE SEQUENCE funcionario_seq START WITH 1 INCREMENT BY 1 NOCACHE;
```

Outputs:

```
--- EXECUTANDO TESTES DA SPRINT 3 ---

--- INICIANDO TESTES DE SUCESSO ---
--- Relatório de Pacientes em formato JSON ---
[
{"nome_paciente":"Ana Silva","idade":"75","nome_abrigo":"Abrigo Esperança","endereco_abrigo":"Rua das Flores, 100"},
{"nome_paciente":"Maria Oliveira","idade":"82","nome_abrigo":"Abrigo Esperança","endereco_abrigo":"Rua das Flores, 100"},
{"nome_paciente":"Carlos Souza","idade":"68","nome_abrigo":"Abrigo Solidário","endereco_abrigo":"Avenida Principal, 250"},
{"nome_paciente":"Fernanda Lima","idade":"60","nome_abrigo":"Abrigo Solidário","endereco_abrigo":"Avenida Principal, 250"},
{"nome_paciente":"João Santos","idade":"91","nome_abrigo":"Abrigo do Povo","endereco_abrigo":"Praça da Liberdade, 15"}
]
--------------------------------------------

Trigger testado com sucesso. Verifique a tabela AUDITORIA.

Resultado (Abrigo 1): Há vagas disponíveis.

--- Relatório de Salários por Abrigo e Cargo ---
Abrigo                         Cargo               Salário
---------------------------------------------------------------------
Abrigo Esperança               ADMINISTRADOR          5,000.00
Abrigo Esperança               FUNCIONARIO            2,500.00
---------------------------------------------------------------------
Subtotal Abrigo Esperança                             7,500.00

Abrigo Solidário               FUNCIONARIO            2,800.00
Abrigo Solidário               VOLUNTARIO             1,500.00
---------------------------------------------------------------------
Subtotal Abrigo Solidário                             4,300.00

Abrigo do Povo                 FUNCIONARIO            1,500.00
---------------------------------------------------------------------
Subtotal Abrigo do Povo                               1,500.00
=====================================================================
TOTAL GERAL                                          13,300.00

--- INICIANDO TESTES DE EXCEÇÃO ---

Teste de Exceção (Função 1): Passando um cursor inválido/fechado.
{"erro": "Cursor inválido ou fechado fornecido."}

Teste de Exceção (Procedimento 1): Nenhum paciente encontrado.
Erro Tratado: Nenhum paciente encontrado para gerar o relatório.

Teste de Exceção (Função 2 - Abrigo 999): ORA-20002: Erro Tratado: Abrigo com ID 999 não encontrado.

Teste de Exceção (Procedimento 2): Nenhum funcionário encontrado.
--- Relatório de Salários por Abrigo e Cargo ---
Abrigo                         Cargo               Salário
---------------------------------------------------------------------
Nenhum funcionário encontrado para o relatório.


Procedimento PL/SQL concluído com sucesso.
```