

# ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

MASTERING RELATIONAL AND NON-RELATIONAL DATABASE



RM: 98078 / Augusto Barcelos Barros

RM: 98570 / Gabriel Souza de Queiroz

RM: 97707 / Lucas Pinheiro de Melo

RM: 98266 / Mel Maia Rodrigues

# Sumário

<b>ANÁLISE E DESENVOLVIMENTO DE SISTEMAS</b>	<b>1</b>
<b>MASTERING RELATIONAL AND NON-RELATIONAL DATABASE</b>	<b>1</b>
<b>Sumário</b>	<b>2</b>
<b>1) Criar 02 procedimentos</b>	<b>3</b>
a) O primeiro procedimento deve fazer join de duas ou mais tabelas relacionais e exibir os dados obtidos das tabelas relacionais no formato JSON. Os dados devem ser transformados do formato relacional para o formato JSON através de uma função desenvolvida pelo grupo	3
b) O segundo procedimento deve ler os dados de uma tabela e, na mesma linha, mostrar o valor de uma coluna da linha atual, o valor dessa mesma coluna na linha anterior e o valor dessa mesma coluna na próxima linha. Caso a linha anterior ou a próxima linha não existir, apresentar a palavra "Vazio". O relatório deve ter, pelo menos, cinco linhas de dados. A tabela e a coluna a ser exibida fica a cargo do grupo	4
<b>2) Desenvolver duas funções</b>	<b>5</b>
a) Uma função deve ler os dados recebidos e transformá-lo para o formato JSON. Não use as funções built-in internas de transformação de e para JSON do banco de dados Oracle	5
b) Uma função deve substituir um do processamento existente em seu projeto no formato Função, como por exemplo verificação da complexidade da senha	6
<b>3) Gatilho</b>	<b>7</b>
	7

# 1) Criar 02 procedimentos

- a) O primeiro procedimento deve fazer join de duas ou mais tabelas relacionais e exibir os dados obtidos das tabelas relacionais no formato JSON. Os dados devem ser transformados do formato relacional para o formato JSON através de uma função desenvolvida pelo grupo

```
CREATE OR REPLACE PROCEDURE PRC_EXIBIR_DADOS_JSON IS
  V_JSON CLOB;
  V_COUNT PLS_INTEGER;
  CURSOR C_DADOS IS
  SELECT
    E.NOME AS EMPRESA_NOME,
    U.NOME AS UNIDADE_NOME,
    U.TELEFONE,
    U.EMAIL
  FROM
    CP1_EMPRESA E
    JOIN CP1_UNIDADE U
    ON E.ID = U.EMPRESA_ID;
  V_CHAVES SYS.ODCIVARCHAR2LIST := SYS.ODCIVARCHAR2LIST('empresa', 'unidade', 'telefone', 'email');
  V_VALORES SYS.ODCIVARCHAR2LIST;
BEGIN
  SELECT
    COUNT(*) INTO V_COUNT
  FROM
    CP1_EMPRESA E
    JOIN CP1_UNIDADE U
    ON E.ID = U.EMPRESA_ID;
  IF V_COUNT = 0 THEN
    RAISE NO_DATA_FOUND;
  END IF;

  FOR REC IN C_DADOS LOOP
    V_VALORES := SYS.ODCIVARCHAR2LIST(REC.EMPRESA_NOME, REC.UNIDADE_NOME, REC.TELEFONE, REC.EMAIL);
    V_JSON := FNC_GERAR_JSON(V_CHAVES, V_VALORES);
    DBMS_OUTPUT.PUT_LINE(V_JSON);
  END LOOP;
EXCEPTION
  WHEN VALUE_ERROR THEN
    DBMS_OUTPUT.PUT_LINE('Erro: Valor inválido.');
```

```
SQL> BEGIN
2   PRC_EXIBIR_DADOS_JSON;
3 END;
4 /
```

```
{"empresa": "Empresa X", "unidade": "Unidade A", "telefone": "11223344",
"email": "unidade.a@email.com"}
{"empresa": "Empresa Y", "unidade": "Unidade B", "telefone": "55443322",
"email": "unidade.b@email.com"}
{"empresa": "Empresa Y", "unidade": "Unidade B", "telefone": "22334455",
"email": "unidade.b@email.com"}
{"empresa": "Empresa Z", "unidade": "Unidade C", "telefone": "33445566",
"email": "unidade.c@email.com"}
{"empresa": "Empresa G", "unidade": "Unidade D", "telefone": "44556677",
"email": "unidade.d@email.com"}
```

PL/SQL procedure successfully completed.

```
SQL> BEGIN
2   PRC_EXIBIR_DADOS_JSON;
3 END;
4 /
```

Nenhum dado encontrado.

- b) O segundo procedimento deve ler os dados de uma tabela e, na mesma linha, mostrar o valor de uma coluna da linha atual, o valor dessa mesma coluna na linha anterior e o valor dessa mesma coluna na próxima linha. Caso a linha anterior ou a próxima linha não existir, apresentar a palavra "Vazio". O relatório deve ter, pelo menos, cinco linhas de dados. A tabela e a coluna a ser exibida fica a cargo do grupo

```
CREATE OR REPLACE PROCEDURE PRC_COMPARAR_LINHAS IS
  CURSOR C_CLIENTES IS
  SELECT
    NOME
  FROM
    CP1_CLIENTE;
  V_NOME_ATUAL CP1_CLIENTE.NOME%TYPE;
  V_NOME_ANTERIOR CP1_CLIENTE.NOME%TYPE;
  V_NOME_PROXIMO CP1_CLIENTE.NOME%TYPE;
BEGIN
  BEGIN
    OPEN C_CLIENTES;
    FETCH C_CLIENTES INTO V_NOME_ATUAL;
    FETCH C_CLIENTES INTO V_NOME_PROXIMO;
    WHILE C_CLIENTES%FOUND LOOP
      IF V_NOME_ANTERIOR IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('Anterior: Vazio, Atual: '
          || V_NOME_ATUAL
          || ', Próximo: '
          || V_NOME_PROXIMO);
      ELSE
        DBMS_OUTPUT.PUT_LINE('Anterior: '
          || V_NOME_ANTERIOR
          || ', Atual: '
          || V_NOME_ATUAL
          || ', Próximo: '
          || V_NOME_PROXIMO);
      END IF;

      V_NOME_ANTERIOR := V_NOME_ATUAL;
      V_NOME_ATUAL := V_NOME_PROXIMO;
      FETCH C_CLIENTES INTO V_NOME_PROXIMO;
    END LOOP;

    IF V_NOME_ATUAL IS NOT NULL THEN
      DBMS_OUTPUT.PUT_LINE('Anterior: '
        || V_NOME_ANTERIOR
        || ', Atual: '
        || V_NOME_ATUAL
        || ', Próximo: Vazio');
    END IF;

    CLOSE C_CLIENTES;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Nenhum dado encontrado.');
```

```
SQL> BEGIN
2  PRC_COMPARAR_LINHAS;
3  END;
4  /
```

Anterior: Vazio, Atual: João da Silva, Próximo: Maria Souza  
Anterior: João da Silva, Atual: Maria Souza, Próximo: João Silva  
Anterior: Maria Souza, Atual: João Silva, Próximo: Ana Oliveira  
Anterior: João Silva, Atual: Ana Oliveira, Próximo: Carlos Pereira  
Anterior: Ana Oliveira, Atual: Carlos Pereira, Próximo: Vazio

PL/SQL procedure successfully completed.

```
SQL> BEGIN
2  PRC_COMPARAR_LINHAS;
3  END;
4  /
```

Nenhum dado encontrado.

## 2) Desenvolver duas funções

- a) Uma função deve ler os dados recebidos e transformá-lo para o formato JSON. Não use as funções built-in internas de transformação de e para JSON do banco de dados Oracle

```
CREATE OR REPLACE FUNCTION FNC_GERAR_JSON(  
  P_CHAVES IN SYS.ODCIVARCHAR2LIST,  
  P_VALORES IN SYS.ODCIVARCHAR2LIST  
) RETURN CLOB IS  
  V_JSON CLOB := '{}';  
  V_COUNT PLS_INTEGER := P_CHAVES.COUNT;  
BEGIN  
  IF P_CHAVES.COUNT != P_VALORES.COUNT THEN  
    RETURN 'Erro: O número de chaves e valores não corresponde.';  
  END IF;  
  
  FOR I IN 1..V_COUNT LOOP  
    V_JSON := V_JSON  
      || ''''  
      || P_CHAVES(I)  
      || ''': ''  
      || P_VALORES(I)  
      || ''';  
  
    IF I < V_COUNT THEN  
      V_JSON := V_JSON  
        || ', '  
    END IF;  
  END LOOP;  
  
  V_JSON := V_JSON  
    || '}';  
  RETURN V_JSON;  
EXCEPTION  
  WHEN VALUE_ERROR THEN  
    RETURN 'Erro: Valor inválido.';  
  WHEN NO_DATA_FOUND THEN  
    RETURN 'Erro: Dados não encontrados.';  
  WHEN OTHERS THEN  
    RETURN 'Erro ao gerar JSON: '  
      || SQLERRM;  
END;  
/
```

```
SQL> BEGIN  
2  DBMS_OUTPUT.PUT_LINE(FNC_GERAR_JSON(SYS.ODCIVARCHAR2LIST('nome', 'idade', 'cidade'), SYS.ODCIVARCHAR2LIST('João', '25', 'São Paulo')));  
3 END;  
4 /
```

```
{"nome": "João", "idade": "25", "cidade": "São Paulo"}
```

```
SQL> DECLARE  
2  v_json CLOB;  
3 BEGIN  
4  v_json := FNC_GERAR_JSON(SYS.ODCIVARCHAR2LIST('nome'), SYS.ODCIVARCHAR2LIST());  
5  DBMS_OUTPUT.PUT_LINE(v_json);  
6 END;  
7 /
```

```
Erro: Número de chaves e valores não correspondem.
```

- b) Uma função deve substituir um do processamento existente em seu projeto no formato Função, como por exemplo verificação da complexidade da senha

```
CREATE OR REPLACE FUNCTION CHECK_PASSWORD_COMPLEXITY(  
  P_PASSWORD IN VARCHAR2  
) RETURN VARCHAR2 IS  
  V_MESSAGE VARCHAR2(200);  
BEGIN  
  
  -- Verifica o comprimento da senha  
  IF LENGTH(P_PASSWORD) < 8 THEN  
    V_MESSAGE := 'A senha é muito curta. Deve ter pelo menos 8 caracteres.';  
  
    -- Verifica se há pelo menos uma letra maiúscula  
  ELSIF NOT REGEXP_LIKE(P_PASSWORD, '[A-Z]') THEN  
    V_MESSAGE := 'A senha deve conter pelo menos uma letra maiúscula.';  
  
    -- Verifica se há pelo menos uma letra minúscula  
  ELSIF NOT REGEXP_LIKE(P_PASSWORD, '[a-z]') THEN  
    V_MESSAGE := 'A senha deve conter pelo menos uma letra minúscula.';  
  
    -- Verifica se há pelo menos um número  
  ELSIF NOT REGEXP_LIKE(P_PASSWORD, '[0-9]') THEN  
    V_MESSAGE := 'A senha deve conter pelo menos um número.';  
  
    -- Se todas as condições forem atendidas  
  ELSE  
    V_MESSAGE := 'A SENHA É VÁLIDA.';  
  END IF;  
  
  RETURN V_MESSAGE;  
EXCEPTION  
  WHEN VALUE_ERROR THEN  
    RETURN 'ERRO: FORMATO DE SENHA INVÁLIDO.';  
  WHEN NO_DATA_FOUND THEN  
    RETURN 'ERRO: DADOS NÃO ENCONTRADOS.';  
  WHEN OTHERS THEN  
    RETURN 'ERRO DESCONHECIDO: '  
    || SQLERRM;  
END;  
/
```

Teste 1: A SENHA É VÁLIDA.

Teste 2: A senha é muito curta. Deve ter pelo menos 8 caracteres.

Teste 3: A senha deve conter pelo menos uma letra maiúscula.

Teste 4: A senha deve conter pelo menos uma letra minúscula.

Teste 5: A senha deve conter pelo menos um número.

PL/SQL procedure successfully completed.

```
SQL> DECLARE  
2  v_message VARCHAR2(200);  
3  BEGIN  
4    v_message := CHECK_PASSWORD_COMPLEXITY(NULL);  
5    DBMS_OUTPUT.PUT_LINE(v_message);  
6  END;  
7  /
```

ERRO: SENHA NÃO INFORMADA.

PL/SQL procedure successfully completed.

### 3) Gatilho

```
CREATE OR REPLACE TRIGGER TRG_AUDITORIA_EMPRESA AFTER
TIPO_OLD,
TELEFONE_OLD,
EMAIL_OLD,
CNPJ_NEW,
NOME_NEW,
TIPO_NEW,
TELEFONE_NEW,
EMAIL_NEW
) VALUES (
AUDITORIA_CP1_EMPRESA_SEQ.NEXTVAL,
'INSERT',
SYSDATE,
USER,
NULL,
NULL,
NULL,
NULL,
NULL,
:NEW.CNPJ,
:NEW.NOME,
:NEW.TIPO,
:NEW.TELEFONE,
:NEW.EMAIL
);
ELSIF UPDATING THEN
INSERT INTO AUDITORIA_CP1_EMPRESA (
ID,
OPERACAO,
DATA_OPERACAO,
USUARIO,
CNPJ_OLD,
NOME_OLD,
TIPO_OLD,
TELEFONE_OLD,
EMAIL_OLD,
CNPJ_NEW,
NOME_NEW,
TIPO_NEW,
TELEFONE_NEW,
EMAIL_NEW
) VALUES (
AUDITORIA_CP1_EMPRESA_SEQ.NEXTVAL,
'UPDATE',
SYSDATE,
USER,
:OLD.CNPJ,
:OLD.NOME,
:OLD.TIPO,
:OLD.TELEFONE,
:OLD.EMAIL,
:NEW.CNPJ,
:NEW.NOME,
:NEW.TIPO,
:NEW.TELEFONE,
:NEW.EMAIL
);
ELSIF DELETING THEN
INSERT INTO AUDITORIA_CP1_EMPRESA (
ID,
OPERACAO,
DATA_OPERACAO,
USUARIO,
CNPJ_OLD,
NOME_OLD,
TIPO_OLD,
TELEFONE_OLD,
EMAIL_OLD,
CNPJ_NEW,
NOME_NEW,
TIPO_NEW,
TELEFONE_NEW,
EMAIL_NEW
) VALUES (
AUDITORIA_CP1_EMPRESA_SEQ.NEXTVAL,
'DELETE',
SYSDATE,
USER,
:OLD.CNPJ,
:OLD.NOME,
:OLD.TIPO,
:OLD.TELEFONE,
:OLD.EMAIL,
NULL,
NULL,
NULL,
NULL,
NULL
);
END IF;
END;
```

```
INSERT INTO CP1_EMPRESA (
CNPJ,
NOME,
TIPO,
TELEFONE,
EMAIL
) VALUES (
'9328434393',
'Empresa 1',
'Tipo 1',
'1234567890',
'empresa@email.com'
);

UPDATE CP1_EMPRESA
SET
NOME = 'Empresa 1 Atualizada'
WHERE
CNPJ = '9328434393';

DELETE FROM CP1_EMPRESA
WHERE
CNPJ = 9328434393;

SELECT
*
FROM
AUDITORIA_CP1_EMPRESA;
```

Max Rows: 500

Columns autosize:

Cell contents

Save as: JSON

	ID	OPER...	DATA O...	USUARIO	CNPJ_OLD	NOME_OLD	TIPO...	TELEFONE...	EMAIL_OLD	CNPJ_NEW
1	4	INSERT	15-SEP-24	RM98078						9328434393
2	5	UPDATE	15-SEP-24	RM98078	9328434393	Empresa 1	Tipo 1	1234567890	empresa@email.com	9328434393
3	6	DELETE	15-SEP-24	RM98078	9328434393	Empresa 1 Atualizada	Tipo 1	1234567890	empresa@email.com	
4	7	INSERT	15-SEP-24	RM98078						9328434393
5	8	UPDATE	15-SEP-24	RM98078	9328434393	Empresa 1	Tipo 1	1234567890	empresa@email.com	9328434393
6	9	DELETE	15-SEP-24	RM98078	9328434393	Empresa 1 Atualizada	Tipo 1	1234567890	empresa@email.com	
7	10	INSERT	15-SEP-24	RM98078						9328434393
8	11	UPDATE	15-SEP-24	RM98078	9328434393	Empresa 1	Tipo 1	1234567890	empresa@email.com	9328434393
9	12	DELETE	15-SEP-24	RM98078	9328434393	Empresa 1 Atualizada	Tipo 1	1234567890	empresa@email.com	
10	13	INSERT	15-SEP-24	RM98078						9328434393