**SCC0241 - Laboratório de Bases de Dados**

**Relatório de execução da aula prática 4**

**Aluno NUSP**

Giancarlo Malfate Caprino 12725025

Adriano da Silva de Carvalho 13692400

**Exercício 1**

**➔ Comentário**

Este espaço destina-se às questões discursivas; apenas algumas questões têm comentários.

**➔ Código**

SELECT \*

FROM DUAL;

**➔ Saída**

Resultados apresentados no terminal da interface SQL; exemplo: status de execução, mensagens de erro, dados (em forma tabular), e outras informações pertinentes. Pode-se enfatizar o texto com negrito ou itálico, mas não underline.

# **Observações:**

* O código será copiado deste documento para execução, portanto, finalize cada linha com um ponto e vírgula;
* Comentários também devem ser colocados junto com o código e iniciados pelos caracteres de comentário “--”;
* O código deve ser enfatizado com a fonte Courier New, de preferência com uma cor diferente de preto;
* Faça a cópia do exemplo acima para cada exercício, mantendo a aparência do relatório ao longo dos exercícios;
* Inicie cada exercício (mas não cada item) em uma página nova usando CTRL+ENTER;
* Itens que não valem nota, não precisam ser reportados;
* Itens curtos podem ser agrupados quando conveniente;
* Objetive a fácil leitura do documento.

**Exercício 1**

**➔ Código**

CREATE OR REPLACE PACKAGE olimpiadas IS

TYPE medalhistas\_t IS RECORD (

ouro VARCHAR2(150),

prata VARCHAR2(150),

bronze VARCHAR2(150)

);

FUNCTION podio(p\_olimpiada L05\_DISPUTA.OLIMPIADA%TYPE,

p\_modalidade L05\_DISPUTA.MODALIDADE%TYPE)

RETURN medalhistas\_t;

END olimpiadas;

/

CREATE OR REPLACE PACKAGE BODY olimpiadas IS

FUNCTION podio(p\_olimpiada L05\_DISPUTA.OLIMPIADA%TYPE,

p\_modalidade L05\_DISPUTA.MODALIDADE%TYPE)

RETURN medalhistas\_t IS

medalhistas medalhistas\_t;

BEGIN

WITH Vitorias AS (

-- Conta o número de vitórias e rankea os países

SELECT VENCEDOR AS Pais,

COUNT(\*) AS Numero\_Vitorias,

RANK() OVER (ORDER BY COUNT(\*) DESC) AS Ranking

FROM L05\_DISPUTA

WHERE OLIMPIADA = p\_olimpiada

AND MODALIDADE = p\_modalidade

GROUP BY VENCEDOR

),

Medalhista\_Ouro AS (

-- Seleciona o país com o maior número de vitórias e, em caso de empate, o mais recente

SELECT Pais

FROM Vitorias

JOIN L05\_DISPUTA D ON Vitorias.Pais = D.VENCEDOR

WHERE Ranking = 1

AND D.OLIMPIADA = p\_olimpiada

AND D.MODALIDADE = p\_modalidade

ORDER BY D.DATA\_HORA DESC

FETCH FIRST ROW ONLY

),

Medalhista\_Prata AS (

-- Seleciona o medalhista de prata

SELECT Pais

FROM Vitorias

WHERE Ranking = 2

FETCH FIRST ROW ONLY

),

Medalhista\_Bronze\_Candidates AS (

-- Seleciona todos os países com o terceiro maior número de vitórias

SELECT Pais

FROM Vitorias

WHERE Ranking = 3

),

Medalhista\_Bronze AS (

-- Seleciona o país mais recente a perder para o medalhista de ouro

SELECT MB.Pais

FROM Medalhista\_Bronze\_Candidates MB

JOIN L05\_DISPUTA D ON MB.Pais IN (D.PAIS1, D.PAIS2)

WHERE D.VENCEDOR = (SELECT Pais FROM Medalhista\_Ouro)

AND D.OLIMPIADA = p\_olimpiada

AND D.MODALIDADE = p\_modalidade

ORDER BY D.DATA\_HORA DESC

FETCH FIRST ROW ONLY

)

SELECT

(SELECT Pais FROM Medalhista\_Ouro) AS Ouro,

(SELECT Pais FROM Medalhista\_Prata) AS Prata,

(SELECT Pais FROM Medalhista\_Bronze) AS Bronze

INTO medalhistas.ouro, medalhistas.prata, medalhistas.bronze

FROM DUAL;

RETURN medalhistas;

END podio;

END olimpiadas;

/

DECLARE

medalhistas olimpiadas.medalhistas\_t;

BEGIN

medalhistas := olimpiadas.podio(2008, 1);

DBMS\_OUTPUT.PUT\_LINE('Ouro: ' || medalhistas.ouro);

DBMS\_OUTPUT.PUT\_LINE('Prata: ' || medalhistas.prata);

DBMS\_OUTPUT.PUT\_LINE('Bronze: ' || medalhistas.bronze);

END;

/

**Exercício 2**

**➔ Código**

ALTER TABLE L05\_DISPUTA

ADD (

PLACAR\_PAIS1 INTEGER,

PLACAR\_PAIS2 INTEGER

);

**Item a**

**➔ Código**

CREATE OR REPLACE PROCEDURE preencher\_placar\_disputas AS

BEGIN

FOR disputa IN (

SELECT ID, PAIS1, PAIS2, VENCEDOR

FROM L05\_DISPUTA

WHERE PLACAR\_PAIS1 IS NULL OR PLACAR\_PAIS2 IS NULL

) LOOP

DECLARE

placar1 NUMBER;

placar2 NUMBER;

BEGIN

-- Gerando placares aleatórios entre 0 e 9

placar1 := TRUNC(DBMS\_RANDOM.VALUE(0, 10));

placar2 := TRUNC(DBMS\_RANDOM.VALUE(0, 10));

-- Ajustando placares de acordo com o vencedor, garantindo que nenhum placar exceda 10

IF disputa.VENCEDOR = disputa.PAIS1 THEN

IF placar1 <= placar2 THEN

placar1 := LEAST(placar2 + 1, 10); -- Garantir que o placar1 não exceda 10

END IF;

ELSIF disputa.VENCEDOR = disputa.PAIS2 THEN

IF placar2 <= placar1 THEN

placar2 := LEAST(placar1 + 1, 10); -- Garantir que o placar2 não exceda 10

END IF;

END IF;

-- Atualizando a tabela com os placares gerados

UPDATE L05\_DISPUTA

SET PLACAR\_PAIS1 = placar1, PLACAR\_PAIS2 = placar2

WHERE ID = disputa.ID;

END;

END LOOP;

COMMIT;

END;

-- Exemplo de execução do procedimento:

BEGIN

preencher\_placar\_disputas;

END;

/

**Item b**

**➔ Código**

CREATE OR REPLACE PROCEDURE listar\_sequencia\_disputas(

p\_olimpiada L05\_DISPUTA.OLIMPIADA%TYPE,

p\_modalidade L05\_DISPUTA.MODALIDADE%TYPE

) IS

medalhistas olimpiadas.medalhistas\_t;

modalidade\_nome L03\_MODALIDADE.ESPORTE%TYPE;

sede\_nome L02\_OLIMPIADA.CIDADE\_SEDE%TYPE;

partida\_numero NUMBER := 1; -- Inicializa o número da partida

BEGIN

-- Obtendo o nome da modalidade

SELECT ESPORTE INTO modalidade\_nome

FROM L03\_MODALIDADE

WHERE ID = p\_modalidade;

-- Obtendo o nome da cidade sede

SELECT CIDADE\_SEDE INTO sede\_nome

FROM L02\_OLIMPIADA

WHERE ANO = p\_olimpiada;

-- Cabeçalho da lista

DBMS\_OUTPUT.PUT\_LINE('Jogos Olímpicos de ' || sede\_nome || ' – Modalidade: ' || modalidade\_nome);

-- Loop para listar cada disputa

FOR disputa\_rec IN (

SELECT D.ID, D.DATA\_HORA, L.NOME AS LOCAL\_NOME,

D.PAIS1, D.PAIS2, D.PLACAR\_PAIS1, D.PLACAR\_PAIS2,

M.NOME AS MIDIA\_NOME

FROM L05\_DISPUTA D

JOIN L04\_LOCAL L ON D.LOCAL = L.ID

LEFT JOIN L11\_TRANSMITE T ON D.ID = T.DISPUTA

LEFT JOIN L10\_MIDIA M ON T.MIDIA = M.ID

WHERE D.OLIMPIADA = p\_olimpiada

AND D.MODALIDADE = p\_modalidade

ORDER BY D.DATA\_HORA

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Partida ' || partida\_numero || ' – ' || disputa\_rec.LOCAL\_NOME ||

' – ' || TO\_CHAR(disputa\_rec.DATA\_HORA, 'DD/MM/YYYY') ||

' – ' || TO\_CHAR(disputa\_rec.DATA\_HORA, 'HH24:MI') ||

' – ' || disputa\_rec.PAIS1 || ' ' || disputa\_rec.PLACAR\_PAIS1 ||

' x ' || disputa\_rec.PLACAR\_PAIS2 || ' ' || disputa\_rec.PAIS2 ||

' – Transmissão: ' || NVL(disputa\_rec.MIDIA\_NOME, 'Não disponível'));

partida\_numero := partida\_numero + 1; -- Incrementa o número da partida

END LOOP;

-- Obtendo e exibindo os medalhistas

medalhistas := olimpiadas.podio(p\_olimpiada, p\_modalidade);

DBMS\_OUTPUT.PUT\_LINE('Medalha de ouro: ' || medalhistas.ouro);

DBMS\_OUTPUT.PUT\_LINE('Medalha de prata: ' || medalhistas.prata);

DBMS\_OUTPUT.PUT\_LINE('Medalha de bronze: ' || medalhistas.bronze);

END listar\_sequencia\_disputas;

/

-- Exemplo de execução do procedimento:

BEGIN

listar\_sequencia\_disputas(2008, 1);

END;

/

**Exercício 3**

**➔ Código**

ALTER TABLE L01\_PAIS

ADD (TOTAL\_OUROS NUMBER DEFAULT 0,

TOTAL\_PRATAS NUMBER DEFAULT 0,

TOTAL\_BRONZES NUMBER DEFAULT 0);

CREATE OR REPLACE PROCEDURE atualizar\_medalhas\_pais IS

medalhistas olimpiadas.medalhistas\_t;

BEGIN

-- Resetando os contadores de medalhas

UPDATE L01\_PAIS

SET TOTAL\_OUROS = 0, TOTAL\_PRATAS = 0, TOTAL\_BRONZES = 0;

-- Loop por cada olimpíada e modalidade

FOR olimpiada\_rec IN (SELECT DISTINCT OLIMPIADA FROM L05\_DISPUTA) LOOP

FOR modalidade\_rec IN (SELECT DISTINCT MODALIDADE FROM L05\_DISPUTA WHERE OLIMPIADA = olimpiada\_rec.OLIMPIADA) LOOP

-- Obtendo os medalhistas

medalhistas := olimpiadas.podio(olimpiada\_rec.OLIMPIADA, modalidade\_rec.MODALIDADE);

-- Atualizando o total de ouros

IF medalhistas.ouro IS NOT NULL THEN

UPDATE L01\_PAIS

SET TOTAL\_OUROS = TOTAL\_OUROS + 1

WHERE NOME = medalhistas.ouro;

END IF;

-- Atualizando o total de pratas

IF medalhistas.prata IS NOT NULL THEN

UPDATE L01\_PAIS

SET TOTAL\_PRATAS = TOTAL\_PRATAS + 1

WHERE NOME = medalhistas.prata;

END IF;

-- Atualizando o total de bronzes

IF medalhistas.bronze IS NOT NULL THEN

UPDATE L01\_PAIS

SET TOTAL\_BRONZES = TOTAL\_BRONZES + 1

WHERE NOME = medalhistas.bronze;

END IF;

END LOOP;

END LOOP;

COMMIT;

END atualizar\_medalhas\_pais;

BEGIN

atualizar\_medalhas\_pais;

END;

/

**Exercício 4**

**Item a**

**➔ Código**

CREATE OR REPLACE PROCEDURE corrigir\_local\_disputa IS

CURSOR disputa\_cursor IS

SELECT D.ID AS disputa\_id, D.LOCAL, D.OLIMPIADA, L.PAIS AS pais\_local, O.PAIS AS pais\_olimpiada

FROM L05\_DISPUTA D

JOIN L04\_LOCAL L ON D.LOCAL = L.ID

JOIN L02\_OLIMPIADA O ON D.OLIMPIADA = O.ANO

WHERE L.PAIS != O.PAIS; -- Identifica inconsistências

TYPE local\_table\_type IS TABLE OF L04\_LOCAL.ID%TYPE;

locais\_validos local\_table\_type;

novo\_local L04\_LOCAL.ID%TYPE;

tentativas NUMBER := 0;

MAX\_TENTATIVAS NUMBER := 10; -- Defina um limite para o número de tentativas

BEGIN

-- Itera sobre as disputas inconsistentes

FOR disputa\_rec IN disputa\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Corrigindo disputa ID: ' || disputa\_rec.disputa\_id);

-- Seleciona locais válidos no país da olimpíada

SELECT L.ID BULK COLLECT INTO locais\_validos

FROM L04\_LOCAL L

WHERE L.PAIS = disputa\_rec.pais\_olimpiada

AND L.ID NOT IN (

SELECT LOCAL

FROM L05\_DISPUTA

WHERE PAIS1 = (SELECT PAIS1 FROM L05\_DISPUTA WHERE ID = disputa\_rec.disputa\_id)

AND PAIS2 = (SELECT PAIS2 FROM L05\_DISPUTA WHERE ID = disputa\_rec.disputa\_id)

AND MODALIDADE = (SELECT MODALIDADE FROM L05\_DISPUTA WHERE ID = disputa\_rec.disputa\_id)

AND OLIMPIADA = (SELECT OLIMPIADA FROM L05\_DISPUTA WHERE ID = disputa\_rec.disputa\_id)

);

-- Atualiza o local da disputa com um local aleatório válido

IF locais\_validos.COUNT > 0 THEN

tentativas := 0;

LOOP

novo\_local := locais\_validos(DBMS\_RANDOM.VALUE(1, locais\_validos.COUNT));

BEGIN

-- Tenta atualizar a disputa

UPDATE L05\_DISPUTA

SET LOCAL = novo\_local

WHERE ID = disputa\_rec.disputa\_id;

DBMS\_OUTPUT.PUT\_LINE('Atualizado disputa ID: ' || disputa\_rec.disputa\_id || ' para novo local: ' || novo\_local);

EXIT; -- Sai do loop se a atualização for bem-sucedida

EXCEPTION

WHEN dup\_val\_on\_index THEN

-- Se houver erro de duplicidade, tenta outro local

tentativas := tentativas + 1;

IF tentativas >= MAX\_TENTATIVAS THEN

-- Se atingir o limite de tentativas, registre um erro ou tome uma ação

DBMS\_OUTPUT.PUT\_LINE('Não foi possível encontrar um local único para a disputa ID: ' || disputa\_rec.disputa\_id);

EXIT;

END IF;

END;

END LOOP;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nenhum local válido encontrado para a disputa ID: ' || disputa\_rec.disputa\_id);

END IF;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Correção concluída.');

END corrigir\_local\_disputa;

BEGIN

corrigir\_local\_disputa;

END;

/

**Item b**

**➔ Código**

CREATE OR REPLACE PROCEDURE verificar\_datas\_disputas AS

BEGIN

FOR disputa IN (

SELECT d.ID, d.OLIMPIADA, o.DATA\_INICIO, o.DATA\_ENCERRAMENTO

FROM L05\_DISPUTA d

JOIN L02\_OLIMPIADA o ON d.OLIMPIADA = o.ANO

WHERE d.DATA\_HORA NOT BETWEEN o.DATA\_INICIO AND o.DATA\_ENCERRAMENTO

) LOOP

-- Selecionar uma data aleatória dentro do período correto

UPDATE L05\_DISPUTA

SET DATA\_HORA = disputa.DATA\_INICIO + TRUNC(DBMS\_RANDOM.VALUE(0, disputa.DATA\_ENCERRAMENTO - disputa.DATA\_INICIO)) +

(DBMS\_RANDOM.VALUE(0, 86399) / 86400) -- Hora aleatória no mesmo dia

WHERE ID = disputa.ID;

END LOOP;

END;

/

BEGIN

verificar\_datas\_disputas;

END;

**Exercício 5**

**Item a**

**➔ Código**

INSERT INTO L06\_ATLETA (PASSAPORTE, NOME, PAIS, GENERO, MODALIDADE) VALUES ('BR12348', 'Juliana Lemos', 'Brasil', 'M', 10);

CREATE OR REPLACE PROCEDURE verificar\_genero\_atletas IS

BEGIN

FOR atleta IN (

SELECT A.PASSAPORTE AS atleta\_id, A.NOME AS atleta\_nome, A.GENERO AS atleta\_genero, M.GENERO AS modalidade\_genero

FROM L06\_ATLETA A

JOIN L03\_MODALIDADE M ON A.MODALIDADE = M.ID

)

LOOP

IF atleta.atleta\_genero != atleta.modalidade\_genero THEN

-- Atualiza o gênero do atleta para o gênero correto da modalidade

UPDATE L06\_ATLETA

SET GENERO = atleta.modalidade\_genero

WHERE PASSAPORTE = atleta.atleta\_id;

-- Exemplo de log de alteração

DBMS\_OUTPUT.PUT\_LINE('Gênero do atleta ' || atleta.atleta\_nome || ' (ID: ' || atleta.atleta\_id || ') corrigido para ' || atleta.modalidade\_genero);

END IF;

END LOOP;

-- Confirma as alterações

COMMIT;

END;

BEGIN

verificar\_genero\_atletas;

END;

/

**Item b**

**➔ Código**

-- Passo 1: Adicionar o atributo Esporte

ALTER TABLE L06\_ATLETA

ADD ESPORTE VARCHAR2(30);

-- Passo 2: Adicionar a chave estrangeira que referencia (Esporte, Genero) na tabela L03\_MODALIDADE

ALTER TABLE L06\_ATLETA

ADD CONSTRAINT KF\_MOD\_EG FOREIGN KEY(ESPORTE, GENERO) REFERENCES L03\_MODALIDADE(ESPORTE, GENERO);

-- Passo 3: Remover a constraint FK\_MOD\_A

ALTER TABLE L06\_ATLETA

DROP CONSTRAINT FK\_MOD\_A;

-- Passo 4: Remover o atributo Modalidade

ALTER TABLE L06\_ATLETA

DROP COLUMN MODALIDADE;

**Item c**

**➔ Código**

CREATE OR REPLACE PROCEDURE preencher\_esporte\_atleta IS

BEGIN

-- Atualiza o campo Esporte na tabela L06\_ATLETA

UPDATE L06\_ATLETA A

SET A.Esporte = (

SELECT MIN(M.Esporte)

FROM L07\_JOGA J

JOIN L05\_DISPUTA D ON J.DISPUTA = D.ID

JOIN L03\_MODALIDADE M ON D.MODALIDADE = M.ID

WHERE J.ATLETA = A.PASSAPORTE

);

END;

BEGIN

preencher\_esporte\_atleta;

END;

/